51CTO.com
技术成就梦想

我 们 只 谈 开 发

开发月刊

Development Monthly

2012年06月
总第015期

北上广之外的IT技术人生

JVM平台开发语言 一直在流行

API正迅速成为Web应用程序粘合剂

developer.51cto.com

目录 2012.06 contents

51CTO.com 技术成就梦想

出版方 :北京无忧创想信息技术有限公司

责任编辑 :彭凡　林师授　张伟　　封面设计 :魏峰

联系方法 :pengfan@51cto.com　010-68476606（分机 8058）

出版日期 :2012 年 6 月 15 日

欢迎广大读者投稿

6月编程语言排行榜：Objective-C

作者 / 张伟

TIOBE 近日公布了 2012 年 6 月份的编程语言排行榜，本月排名前两位的仍是 C 和 Java，呈较好上升趋势的依旧是 Objective-C。依旧排在第四的位置，并且近一步拉近了与 C++ 的距离。C++ 虽依旧排在老三的位置，相比之下，还是有少量的减少。由此可见 Objective-C 具有很大的发展空间。

Objective-C 流行的主要原因可能是它是唯一一种可以为 iPhone 和 iPad 等基于 iOS 系统编程的语言。但单从编程语言的角度来看，诞生于 1986 年的 Object-c 并没有表现出多少新意。然而 Objective-C 却又是非常 “实际” 的语言。

大家先看 2012 年 6 月编程语言排行榜榜单

Position Jun 2012	Position Jun 2011	Delta in Position	Programming Language	Ratings Jun 2012	Delta Jun 2011	Status
1	2	↑	C	17.725%	+1.45%	A
2	1	↓	Java	16.265%	-2.32%	A
3	3	=	C++	9.358%	-0.47%	A
4	7	↑↑↑	Objective-C	9.094%	+4.66%	A
5	4	↓	C#	7.026%	+0.18%	A
6	6	=	(Visual) Basic	6.047%	+1.32%	A
7	5	↓↓	PHP	5.287%	-1.31%	A
8	8	=	Python	3.848%	-0.05%	A
9	9	=	Perl	2.221%	-0.09%	A
10	12	↑↑	Ruby	1.683%	+0.20%	A
11	11	=	JavaScript	1.474%	-0.03%	A
12	29	↑↑↑↑↑↑↑↑↑	Visual Basic .NET	1.216%	+0.78%	A
13	13	=	Delphi/Object Pascal	1.150%	+0.08%	A
14	14	=	Lisp	0.986%	+0.05%	A
15	21	↑↑↑↑↑↑	Logo	0.860%	+0.31%	A-
16	15	↓	Pascal	0.844%	+0.11%	A
17	17	=	Transact-SQL	0.705%	+0.05%	A
18	19	↑	Ada	0.681%	+0.08%	B
19	22	↑↑↑	PL/SQL	0.637%	+0.13%	A-
20	10	↓↓↓↓↓↓↓↓↓↓	Lua	0.635%	-1.40%	B

Objective-C 是非常“实际”的语言

它使用一个用 C 写成、很小的运行库，只会令应用程序的大小增加很小，和大部分 OO 系统使用极大的 VM 执行时间会取代了整个系统的运作相反。

ObjC 写成的程序通常不会比其原始码大很多。而其函式库（通常没附在软件发行本）亦和 Smalltalk 系统要使用极大的内存来开启一个窗口的情况相反。

Objective-C 的最初版本并不支持垃圾回收。在当时这是争论的焦点之一，很多人考虑到 Smalltalk 回收时有漫长的 “死亡时间”，令整个系统失去功用。Objective-C 为避免此问题才不拥有这个功能。

虽然某些第三方版本已加入这个功能（尤是 GNUstep），Apple 在其 Mac OS X 10.3 中仍未引入这个功能。不过令人欣慰的是在 Apple 发布的 xCode4 中已经支持自动释放啦，我不敢冒昧地说那是垃圾回收，因为两者机制不同，在 xCode4 中的自动释放，也就是 ARC(Automatic Reference Counting) 机制，是不需要用户手动去 Release 一个对象，而是在编译期间，编译器会自动帮你添加那些以前你经常写的 [NSObject release]。

虽然 Objective-C 是 C 的母集，但它也不视 C 的基本型别为第一级的对象。

和 C++ 不同，Objective-C 不支援运算子多载（它不支持 ad-hoc 多型）。

6 月编程语言排行榜 :Objective-C 非常“ 实际 ”的语言 II

亦与 C++ 不同,但和 Java 相同,Objective-C 只容许对象继承一个类别(不设多重继承)。Categories 和 protocols 不但可以提供很多多重继承的好处,而且没有很多缺点,例如额外执行时间过重和二进制不兼容。

Objective-C 和 C++ 的比较

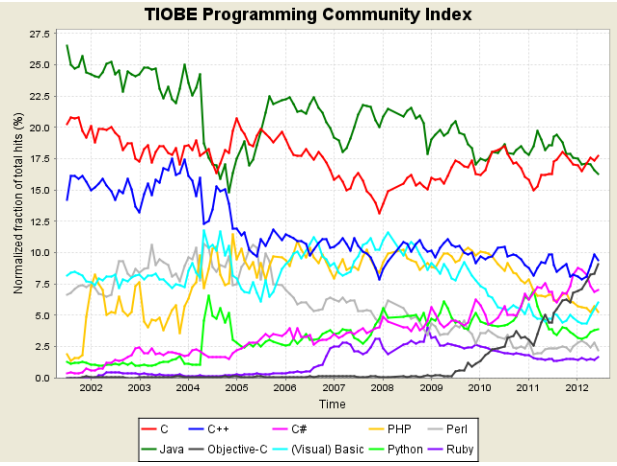
单一继承: Objective-C 不支持多重继承,(同 Java 和 Smalltalk),而 C++ 语言支持多重继承。

动态: Objective-C 是动态定型(dynamicaly typed) 所以它的类库比 C++ 要容易操作。Objective-C 在运行时可以允许根据字符串名字来访问方法和类,还可以动态连接和添加类。

C++ 跟从面向对象编程里的 Simula 67(一种早期 OO 语言) 学派, 而 Objecive-C 属于 Smalltalk 学派。在 C++ 里,对象的静态类型决定你是否可以发送消息给它,而对 Objecive-C 来说,由动态类型来决定。

Simula 67 学派更安全,因为大部分错误可以在编译时查出。 而 Smalltalk 学派更灵活,比如一些 Smalltalk 看来无误的程序拿到 Simualr 67 那里就无法通过。

前 10 名编程语言走势图

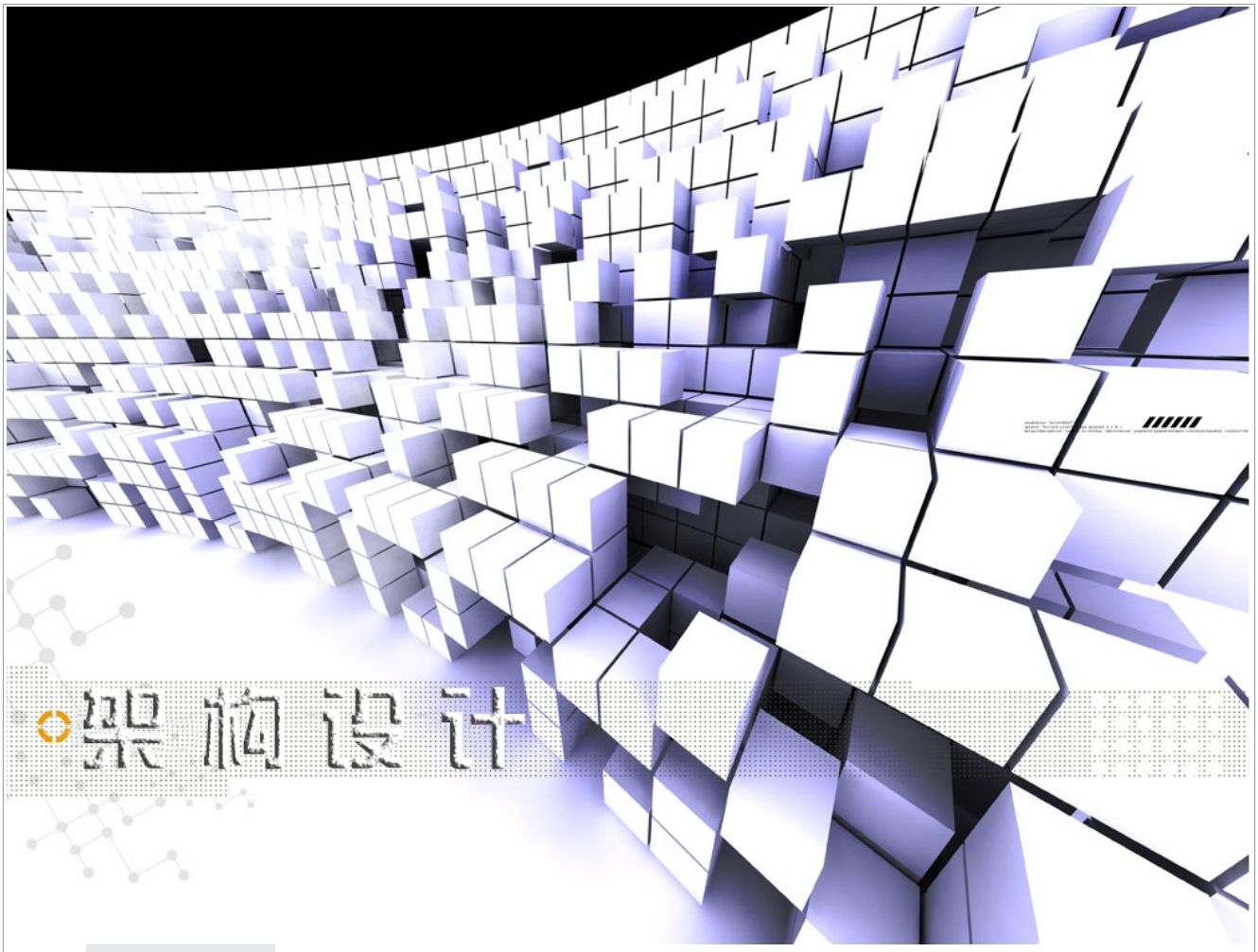


20 到 50 名语言排行

Position	Programming Language	Ratings
21	Bash	0.622%
22	MATLAB	0.563%
23	Assembly	0.538%
24	SAS	0.534%
25	Haskell	0.532%
26	ABAP	0.476%
27	RPG (OS/400)	0.470%
28	COBOL	0.463%
29	Fortran	0.451%
30	R	0.438%
31	Scheme	0.421%
32	Scratch	0.339%
33	D	0.339%
34	Prolog	0.329%
35	NXT-G	0.329%
36	(Visual) FoxPro	0.265%
37	Erlang	0.257%
38	Awk	0.257%
39	Smalltalk	0.247%
40	APL	0.243%
41	Scala	0.234%
42	Forth	0.230%
43	ML	0.224%
44	Ladder Logic	0.209%
45	Max/MSP	0.199%
46	Alice	0.196%
47	ActionScript	0.189%
48	Algol	0.169%
49	CFML	0.165%
50	PowerShell	0.151%

下面是第 50 到 100 的编程语言排名

ABC, bc, Boo, C shell, cg, CHILL, CL (OS/400), Clean, Clojure, Cobra, cT, Curl, Dylan, Eiffel, Euphoria, F#, Factor, Gambas, Go, Groovy, Icon, Informix-4GL, J, JavaFX Script, JScript.NET, LabVIEW, Lingo, Magic, Modula-2, MUMPS, NATURAL, Oberon, OCaml, Occam, OpenCL, OpenEdge ABL, Oz, PL/I, Q, REXX, S, SPARK, Standard ML, SuperCollider, Tcl, VBScript, VHDL, X10, xBase, XSLT



在软件开发过程中需求是不停的变化，随着客户对系统的认识，和现有开发功能和软件的认识，也许以开始他提出的需求就是背离的。记得网上有一句笑话，师说需求变化的：

程序员 XX 遭遇车祸成植物人，医生说活下来的希望只有万分之一，唤醒更为渺茫。可他的 Lead 和亲人没有放弃，他们根据 XX 工作如命的作风，每天都在他身边念："XX，需求又改了，该干活了，你快来呀！"，奇迹终于发生了，XX 醒来了，第一句话："需求又改了"。

在设计和架构中，凡事无绝对，作为架构师或者项目负责人你必须永远的清晰认识到没有完美的架构和设计，没有万能的软件。只存在当前环境，需求方案，团队人员素质，物理环境，安全等综合因素下的合适方案，由于总总原因你的解决方案可能不是某一个单一因素下的最优解。站在这个位置你需要做的是找到这个综合下的最优解权衡。

若你关注这些，那么本专题不可错过！本系列专题文章结合实例帮助开发者了解和优化网站的性能。

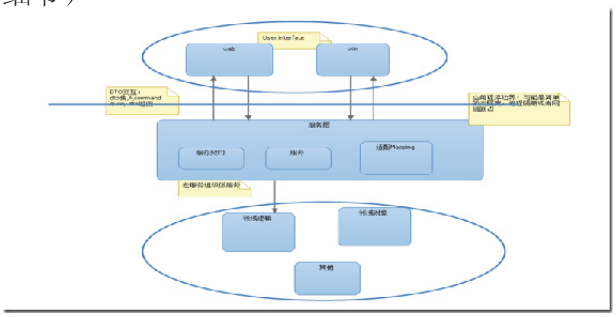
51CTO 开发频道语

架构设计中服务层的简单理解

在 ddd 设计中我们经常会提到服务层，服务层是什么？职责是什么？有什么好处？下面作者将详细解读这几个问题。先看简单的层次图（注：这里并没有考虑其他多余的领域逻辑数据层存储，或者 UOW 这些细节）

在 ddd 设计中我们经常会提到服务层，服务层是什么？职责是什么？有什么好处？

先看简单的层次图（注：这里并没有考虑其他多余的领域逻辑数据层存储，或者 UOW 这些细节）



我的理解是服务层是处于我的应用程序业务层和表现层之间的应用程序边界，边界可能是很薄的一层类设计或者是分布式服务网络跃点。它是一个与技术无关的名词。由表现层直接调用，契约，执行命令（修改状态（CUD））或者是查询返回 dto（数据迁移对象）（cms，命令 – 查询分离）。他对业务逻辑层接口很清楚，组织业务逻辑 微服务形成宏服务，适配表现层。

这里谈到宏服务和微服务，宏服务有一些列粗粒度的服务组成。用户的一次操作 usecase，比如电子商务下单，CreateOrder 就是一个宏服务，而不是下单中的细粒度的商品库存检查，订单合法性等。而与之对应的微服务（有时也叫应用程序服务），则表现为问题领域逻辑细节，就如上面的库存检查和合法性检查这些细粒度的服务。宏服务是由一个或者多个微服务组成，有时我们的 usecase 逻辑很简单服务层仅由单一微服务组成，变现为很简单的几句微服务调用。

服务层的职责：

1：在面软件开发不管是结构化编程（sp）还是面向对象编程（oop）我们一直都强调高内聚低耦合，分离关注点（soc）。服务层处于应用程序和业务层之间，应用边界，使得两次直接解耦，利用第三个对象破坏两对象直接的依赖，并转化适配领域对象（do）和试图对象（vo）的差异。

2：服务层隐藏了业务逻辑层的细节，其内部需要组织业务微服务，提供更宏观，面向表现层的服务逻辑，利用契约接口暴露，包装。系统所有的交互都是从表现层进入。

目前流行 SOA 架构，提供了一种分布式服务架构，以服务为关注点，提高服务和业务逻辑的重用，但是这里说的服务并不是特定的技术 wcf 或者 webservice，服务同时候可能是一次规定契约的一些列粗粒度组织的类组成。但是利用 SOA 或者 MTS 建立服务会让我们的服务得到跟多的附加优势，例如安全，事物，日志，扩展性的提升。

架构设计中服务层的简单理解 II

服务层带来的优势：如上所述服务层为表现层提供的同一的接口契约和入口。让我们的业务层可以关注与实现问题领域逻辑，问题领域实际需求。组织微服务避免太多的细粒度服务的调用充斥在我们的项目表现层和问题领域中，过多的交互。如果采用 soa 等服务领域可以让我们的应用程序轻易的跨过应用程序边界和网络跃点。但是需要付出一点的性能代价。

数据迁移对象（dto）就是携带数据穿过应用程序边界的对象，减少数据的交互次数，常常我们将其作为值对象，只是一组简单的 get，set 属性组成，不存在行为操作，仅仅为数据的载体。在领域设计中 dto 是一个很重要的模式，不是我们所有的领域对象都能轻松的到达表现层，仅仅表现层和领域层部署在同一物理位置。如果需要穿过网络跃点或者进程边界，因为领域对象使我们的业务的核心存在很多的自然世界的关系，依赖，甚至可能存在循环依赖比如电商用户和订单，用户用户一组订单的集合，而每个订单都指向一个特定的用户，我们就必须破换掉这种循环依赖，才可能使其可序列化，穿过跃点。其次我们的领域对象往往都是一堆领域富对象，存在大量数据，很多时候我们的场景并不需要全部的数据信息。有了 dto 的存在就能很好的解决这些问题，是的我们的项目变得 simple（keep it simple，Stupid。 KISS 原则）。

但是与此同时 dto 存在会为我们带来一些额外的复杂度，我们必须有一层 do 到 dto 的映射适配层。

理论上完美的设计我们需要为每一个应用定义一个 dto，但是在一个复杂的系统中我们可能存在很多的领域对象，加入 500 个 do，每个 do 一般都会存在多个 dto，这将一个增加一个庞大的集合和 mapping 逻辑，对于维护也存在不小的挑战。在软件领域存在一句话就是 bug 的数量随着代码量增加，代码量增加需要测试点也随着增加。除非我们必须跨越应用程序网络跃点边界，我觉得否则我们也可以存在一些简单 do 的直接使用。根据世界项目，情形由我们的架构师决定。■

架构设计：数据访问层简述

作者 / 破狼

在前面简单描述了下服务层，SOA 面向服务架构，架构设计 – 业务逻辑层，以及一些面向设计原则理解和软件架构设计箴言。这篇博客我们将继续进入我们的下一层：数据访问层。无论你用的是什么开发模式或者是业务模式，到最后最必须具有持久化机制，持久化到持久化介质，并能对数据进行读取和写入 CRUD。这就是数据访问层。你可能是利用 xml 等文件格式磁盘存储，常用的关系数据库存储，或者 NoSql(not only sql) 的内存存储或文档存储等等存储介质。而这里我只关心关系数据库存储。

数据层需要提供的职责有：

1：CRUD 服务。作为唯一可以与存储介质交互的中间层出现，负责业务对象的增加，修改，删除，加载。

2：查询服务。这不同于 CRUD 中的 R（read），read 倾向于的单个对象，元组。而这里的查询针对复杂查询，比如一个国内电商的客户为四川的订单。这里会涉及仓储层。所谓仓储模式指的是一个提供业务对象查询的类，他隐藏了数据查询的解析步骤，封装 sql 解析逻辑。

3：事务管理。这里所说的是业务事务，在一个应用系统中每次请求都会产生多次的多数据对象的新增，修改，删除操作。如果我们每次都依次代开数据库连接，准备数据包，操作数据库，关闭数据连接。这些将会给我们带来很多不必要的性能开销。数据库管理员经常会要求“尽量少的与数据库交互”，这也必须成为我们的开发原则。更好的操作是我们在内存中建立一个和数据仓库，维护变化的对象，在业务操作完成一次性提交到数据存储介质，提供业务事务。业务事务有个很好听的名字工作单元（UOW），在微软给我们提供的 DataSet，orm 框架都回必须存在业务事务。

4：并发处理。UOW 应避免业务数据连接的多次提交打开而出现，但在内存离线操作，这就可能导致数据一致性问题。在多用户的环境，对数据并发处理需要制定一个策略。一般我们会采用乐观并发处理：用户可以任意的离线修改，在修改更新时候检查对象是否被修改，如果被修改者本次更新失败。简单的说就是防止丢失修改。防止丢失修改，我们可以采用 where 加上一系列原值，或者加上修改时间戳或者版本号标记。同时还有许多其他的并发解决模式，但乐观并发锁用到更普遍。

5：数据上下文：整和所有职责。在数据访问层概念职责都会有一个共同的暴露给外部的接口。我们需要一个高层次的组件，来同一提供对数据存储介质的访问操作。，同一访问数据库 CRUD，事务，并发服务的高层次类，叫做数据上下文（Context）。EF 中的 ObjectContext，NHibernate 的 session，linq to sql 的 DataContext 等等。数据访问层的一些概念（这里不会是全部，仅一些个人觉得重要的概念）：

1： 数据映射器：将内存中修改的对象提交至存储介质，则需要要映射逻辑来完成，数据映射器就是就是一个实现将某种类型的业务对象持久

架构设计 :数据访问层简述 II

化的类(数据映射器模式定义如《P of EAA》)。

2: 仓储层(Repository): 在上面提到: 所谓仓储模式指的是一个提供业务对象查询的类,他隐藏了数据查询的解析步骤,封装 sql 解析逻辑。在面向对象的世界里我们用对象进行查询,返回结果为对象集。这里的查询可能是从数据库,或者来至缓存,这取决你的策略,你仓储层的实现。

3: 工作单元(UOW): Martin Fowler《P of EAA》 定义: 工作单元记录在业务事务过程中对数据库有影响的所有变化。操作结束后,作为一种结果,工作单元了解所有需要对数据库做的改变。在上面第 3 点业务事务讲的差不多,这里不是累述。

4: 标示映射(Identity Map): 其作用在于: 便于跟踪业务对象,调用者在一个业务事务中使用的是同一个实例,而不是每次执行产生一个新的对象。表示映射为一个散列表存储(散列具有快速定位 O(1))。类似于缓存的实现方式,保证了在同一个业务事务数据上下文引用修改同一个业务对象。但绝不同于缓存。从持续时间来说,标示映射生命周期为业务事务内。实现上等同于数据上下文期。但比起缓存来说其周期太短,根本不能对性能有多大的改善。缓存更重要的命中率,而标示映射保证同一数据上下文采用修改同一个业务对象的引用。

5: 乐观并发锁: 在上面也曾提到,其保证离线操作数据的对数据一致性的冲突解决方法。首先乐观在于允许离线操作,容忍冲突,防止数据丢失修改,可利用原读取数据值得 where 条件或者时间戳,版本号解决。

6: 延时加载: 对象并不是一次性加载完成,而是按照需求多次加载数据,到用时加载。业务对象太多关联,数据量太多余庞大,而我们每次业务事务需要操作的对象都只会是部分,不需要太多的数据对象。同事业务对象中还存在循环引用,这样不适于对象整体的一次性加载。延时加载提供了优化,意图在 “尽可能的少加载,并按需加载,只加载需要的数据部分”。

7: 持久化透明对象(PI 或 POCO): 当对象模型不存在任何外部依赖,特别是对于数据访问层的依赖,那么这个模型就是持久化透明的,POCO。一个 POCO 的对象不需要继承至某个特定的类,实现特定的接口,或提供专门的构造函数。一个非持久化透明的对象这以为者存在外部的依赖,而我们更喜欢领域对象只是一个简单额 c# 类,可以在持久化层等独立切换。这就导致实现的时候我们无法很直接的跟踪业务对象,这就是面向方面编程(AOP)或者代理模式的大显身手。可惜 AOP 在 .net 中不是那么直接,很多 ORM 框架如 NHibernate 之类的利用代理模式 Emit 动态注入 IL 实现跟踪,添加新的行为。所以 NHibernate 中要求领域对象的所有字段属性方法都必须是虚方法可重写的。:

8: CQRS (Command Query Responsibility Segregation, 命令查询职责分离): CQRS 是在 DDD 的实践中引入 CQS 理论而出现的一种体系结构模式,命令和查询被分离。具体可以参 Martin Fowler 的 CQRS 文章:

http://martinfowler.com/bliki/CQRS.html■

架构设计：逻辑层vs物理层

Layer 和 Tier 都是层，但是他们所表现的含义不同，Tier 指的是软件系统中物理上的软件和硬件，具体指部署在某服务器上，而 Layer（逻辑层）指软件系统中完成特定功能的逻辑模块，逻辑概念。Layer 是逻辑上 组织代码的形式。

Layer 和 Tier 都是层，但是他们所表现的含义不同，Tier 指的是软件系统中物理上的软件和硬件，具体指部署在某服务器上，而 Layer（逻辑层）指软件系统中完成特定功能的逻辑模块，逻辑概念。

Layer 是逻辑上 组织代码的形式。比如逻辑分层中表现层，服务层，业务层，领域层，他们是软件功能来划分的。并不指代部署在那台具体的服务器上或者，物理位置。

Tier 这指代码运行部署的具体位置，是一个物理层次上的划为，Tier 就是指逻辑层 Layer 具体的运行位置。所以逻辑层可以部署或者迁移在不同物理层，一个物理层可以部署运行多个逻辑层。

从 Layer 和 Tier 就会延伸到逻辑架构和物理架构。我们一个逻辑分层（N-Layer）的部署运行环境可以在一台或者是多台服务器，由于物理环境的多样性，逻辑层次的部署也具有多样性。这就需要我们必须了解物理架构和逻辑架构。

大多数情况下我们所说的 N 层应用系统指的是物理模型，具体模块的分布物理位置。客户端，服务层，逻辑层，数据库服务器，与我们的逻辑模型之间并不是一对一的关系。逻辑上的分层架构与物理位置上的服务器数量和网络边界多少无关，逻辑架构层次只与我们的功能划分相关，是按照功能划分。经典的 3-Layer 架构：表现层，业务层，数据访问层，他们可能运行在同一物理位置上。也可以是 3 台计算机上，这并不是逻辑架构所关注的。逻辑层次和物理分层数量关系为：逻辑层数必须不小于物理层数，因为一个物理层可以部署一个或者多个逻辑层次，逻辑层次只能迁移在不同的物理环境。

逻辑层次的架构能帮助我们解决逻辑耦合，达到灵活配置，迁移。

一个良好的逻辑分层可以带来：

逻辑组织代码

易于维护

代码更好的重用

更好的团队开发体验

代码逻辑的清晰度

一个良好的物理架构可以带来：

性能的提升

可伸缩性

容错性

安全性

逻辑层次越多会影响程序运行的性能，但代码层次的低耦合，松散化，是需要架构师的权衡的，我觉得一般应用程序的瓶颈并不在这里。■

编者按

今天是 22 岁的最后一天。几个月前，我从沃顿商学院毕业，用文凭上“ 最高荣誉毕业 ”的标签安抚了已经年过半百的老妈，然后转头辞去了毕业后的第一份工作，跟一家很受尊敬的公司、还有 150 万的年薪道了别。

给明年依然年轻的我们:道别150万年薪,开始盒饭生活

今天是 22 岁的最后一天。几个月前，我从沃顿商学院毕业，用文凭上“最高荣誉毕业”的标签安抚了已经年过半百的老妈，然后转头辞去了毕业后的第一份工作，跟一家很受尊敬的公司、还有 150 万的年薪道了别，回到了上海，加入了“刚毕业就失业”俱乐部，开始了一天三顿盒饭的新生活，中间许多精彩剧情暂时略过。

我肯定不是第一个做过这样事的人，也肯定不会是最后一个。所以在说自己的一些有趣故事前，我想借用大家（包括 30 岁甚至 40 岁以上的朋友）的一点时间和一点平和的心态，和大家分享过去一年以来一直没说的一些话。所以前两部说的是对于一些一直困扰着我们的关键词的理解和体会。他们是：欲望、外界、标签、天才、时间、经历、人生目标、后悔、和现实。

这可能会是一篇科普文，也可能会是一篇长篇小说，但我不想这篇文章变成一篇励志文，大家都审美疲劳了。所以我想忽略阳春白雪，尽管信息量很大，但是至少说一些实实在在的经验和故事，说一些效果立竿见影的观点，再说说活捉林志玲什么的，总之让大家多看一点就多获得一点实际的价值。

第一部 :那些最容易被理解错误的事

关于欲望

这些是我们内心里和人生理想一样真实的东西：学历、工作、房、车、财富、以及爱。我们每个人都愿意为了这些欲望去付出，无论付出的是汗水、鲜血、还是身体健康、又或是其它你懂的。尽管我们付出的方式可能不被社会主流认同、可能没那么具有有戏剧性，但你和我、北大图书馆里的学生和网吧中奋斗的少年、职场杜拉拉和夜场里跳舞的小姐、韩寒和芙蓉凤姐（韩少躺着也中枪 -_-），我们谁没有为了一个目标连续熬夜奋斗过呢？我们谁没有为了得到一样东西而撕心裂肺地付出过呢？谁没有过那种拼命得快受不了的感觉呢？所以我们最不缺励志的故事，因为我们每个人都是付出领域的专家。

真正的问题是，当我们跑得越快，越是无法考虑我们是否在朝着正确的方向奔跑。

北野武讲过一个很有趣的故事。他说他没出名之前想有一天有了钱，一定要开跑车，吃高档餐厅，跟女人们睡觉。而真正功成名就的时候，他发现开保时捷的感觉并没有那么好，因为“看不到自己开保时捷的样子”。结果他就让朋友开，自己打个出租车，在后面跟着，还对出租司机说：看，那是我的车。

我想说，过去几年里我认识的、深交的、共事过的所有人，包括身边一批又一批二十出头收入一百多万的金融朋友、三十岁左右收入几百万的前辈朋友、以及简历金碧辉煌得已经不在乎收入

给明年依然年轻的我们：道别 150 万年薪，开始盒饭生活 II

的大 BOSS 以及我的经历告诉我两件事：

一，顶级学校的文凭、顶级公司的工作、顶级的收入、顶级的房、顶级的车、顶级的声望，这些都无法满足人类。

二，无论是通过爸妈，通过运气，还是通过奋斗得到这些顶级的东西，人类都不会得到更多的幸福感。

接着北野武的故事说下去。想象一下：你今天骑在一辆助动车上，一个小山村来的年轻人经过，说你的车好帅，你不会有任何的满足感。十几年的奋斗后，你坐在一辆你今天都叫不出型号的保时捷的驾驶位上，一个路人经过，说你的车好帅，相信我，你也不会有任何的满足感。你不在乎他，就像你今天不在说你助动车帅的人。你的视角在变。每当我们考虑许多年后能够取得的成就，我们总是习惯站在今天的角度去衡量幸福感和满足感。你今天的视角只是错觉，却让你相信自己的目标是正确的。这是我们最容易跑错方向时。

人类的需求是很奇特的。我们吃第一个面包的时候的幸福感，和我们吃第一千个面包的时候的幸福感，是差不多的，前者甚至比后者还多一些。同样的感觉适用于我们赚到的第一笔一万元和第一笔一千万元，第一辆十万的车和第一辆一千万的车，第一个女孩和第十个女人，第一个男生和第十个男人。“生理需求、安全需求、归属与爱的需求、尊重的需求和自我实现的需求” -- 在著名的马斯洛五大需求中，你从任意一个细分需求里获得的幸福感只能有那么多。

我们清楚地知道快感和幸福感的不同，我们也知道欲望和需求是两个东西（你从来没有听说过“马斯洛五大欲望”对不对？），但是我们的不幸福却是因为不小心把快感当成了幸福感，把欲望当成了需求，而这就是因为我们常站在现在的视角去想象未来的感受。事实是，就好像我们不需要很多的面包一样，我们不需要很多的财富，不需要很多的爱。因为他们很难给你带来更多快乐。当然，我们也不需要去拔高理想和自由的重要性。你可以尝试着停下来思考一下，这五种需求是否真的有高低之分，思考一下，是否连最贫穷最饥饿的人们，都一直在生活中同时追求着这五个高低层次的需求。你会发现其实这五种需求一样真实，离你一样近，也一样远。然后你需要找到一个可以同时实现这五种需求的平衡点。这个平衡点就是只属于你的奔跑方向。这篇文章会实实在在地帮你找到这个方向。但在这之前，我们先谈一些别的。

关于外界

外界带给我们生活最大的影响是嫉妒和比较。

我们一直高估了嫉妒。举个例子，没有人嫉妒雷帝 Gaga。雷帝 Gaga 应该要比我们都更有名、更有钱、坐更好的车、住更大的房子，比我们更随心所欲，而且也比我们更有才华。但你不嫉妒她，对么？我们没有人嫉妒雷帝 Gaga-- 因为她实在是太雷了。她奇怪得让我们完全不能把我们自己跟她联系在一起，所以我们在名利和才华面前没有自卑，也没有嫉妒，更没有仇恨。反而，我们会去思考，觉得她挺有趣的，挺发人深省的，不是么？本文其他部分，请参考原文，链接：

http://developer.51cto.com/art/201110/299750.htm ■

给明年依然年轻的我们：道别 150 万年薪，开始盒饭生活 III

所以当你见到好事情发生在了那个他或者那个她身上，嫉妒的小火苗在你心中扑哧扑哧的时候，不如把 TA 当成那个很奇怪的雷帝 Gaga 吧。因为这样的时候，我们就会懂得抛开个人的杂念，去真正思考别人的亮点。

至于比较（Social Comparison），我们可以选择努力向那个绩点 4.0 的同学看齐，努力向那个年薪几十万的旧识看齐，努力向那个不断得到提拔的同事看齐。或者，我们也可以选择看看外面更大的世界，那些和我们一样年轻的人们。看上去像是有 30 岁阅历的阿呆 Adele，19 岁时出了张白金专辑《19》，21 岁时出了全销量 1200 万张的专辑《21》，拿了两座格莱美。她出生于 1988 年。眼神和心态似乎已经像中年人那样淡定的杜兰特和德里克罗斯，两个毫无疑问的超级球星，他们也出生于 1988 年。如果你喜欢实用一点的，那么 iPhone 上用户量最大的个人开发第三方浏览器猛犸浏览器的开发者，是一个 1992 年出生的北京少年。如果你的视线中有一个世界舞台，那么你会看到上面的人物已经越来越接近你的年龄。

我们不需要去看齐，我们只需要去“看”。去看到这个世界除了你现在正处在的那个若干平米的封闭空间以外，还有许多许多精彩的事正在发生。当你发现这个世界的深度和广度，你就会发现你跟你身边的那些“同类人”根本没什么好比的。这个世界太大了。你不是你自己的标杆，别人也不是。谁都不是你的标杆，这是一个没有标杆的时代。

我们要做的是试着不去嫉妒，不去比较，更不要批判，但要试着去观察、去倾听，然后去思考、去沉淀、去让所有外界的信息在你大脑里经历一个长时间的处理过程。在你的大脑还没有沉淀出你自己对一件事的观点前，不要发表观点，不要给出你的定论。我们可以不断在大脑中质疑我们所看到的、听到的，我们可以不断挑战自己的想法、挑战任何理所当然的存在，只要我们保证我们的大脑一直在思考，独立地思考。要记得，你和世界上所有人都不一样。

关于标签

“牛逼”是过去几年里笔者听到的比较多的一个形容词。当我们喜欢的人称赞我们的时候，我们总是 P 颠 P 颠的。在这里为自己开脱一下，觉得这挺好，说明活得挺真实。

但笔者想用一个很好的朋友（自己来认领）去年当着我面描述我听的原话，来翻译一下这个已经被用得和“帅哥”“美女”一样烂俗的词。她说，“你想太多了（这是她一贯的开场白）。你只是有很多很牛的标签 -- 上海中学、沃顿商学院、最高荣誉、黑石的全职 Offer、百万年薪。至于你本身么，牛不牛就说不清楚了。”

这个故事告诉我们：一、“牛”和“帅哥”“美女”一样，是一种打招呼的方式，二、“牛”的从来都是那些标签，那些改变了金融产业的企业，那些通过培养人才改变了世界的学校，那些定义了时尚的品牌。虽然我无意改变大家打招呼的方式，但对于还没奔到三的人类来说，“高档”“精英”“牛逼”其实不如“做得不错”或者“挺有意思的”来得更实在。当然，等奔到了三，我们就更不想用这些词了。

如果你曾经或者将来获得了任何标签，……

本文未完，阅读全文请链接：

http://developer.51cto.com/art/201205/337301.htm ■

JVM平台开发语言 一直在流行

作者 / 小林

关于JVM平台开发语言一直是Java开发者茶余饭后的一个热门话题，JVM作为一种机制用以提供Java应用在不同的环境，其他平台以及不同硬件上的可移植性，而基于JVM平台的编程语言也是不断的在增加。从动态语言如Groovy、JRuby, Jython到静态型Scala语言， JVM也正成为一个多元化的平台，开发人员可以充分利用不同语言来满足各自不同的需求。

从本月的编程语言排行榜上我们可以看出，本次最受欢迎的5种编程语言中3种就属基于JVM平台开发的编程语言。这也意味着开发者对JVM平台开发语言的关注、使用也不断的增加。为什么JVM的编程语言会如此受到开发者的青睐？下面我们从JVM平台的3点特性进行分析。

动态语言支持

从SE 6开始就是对动态语言的支持，动态语言逐渐成为许多Java开发者应用开发日常工作的一部分，并常常用于原型开发或用来提高开发速度。为了快速得到更大规模的Java应用程序，人们一直在使用动态语言胶合程序部件。

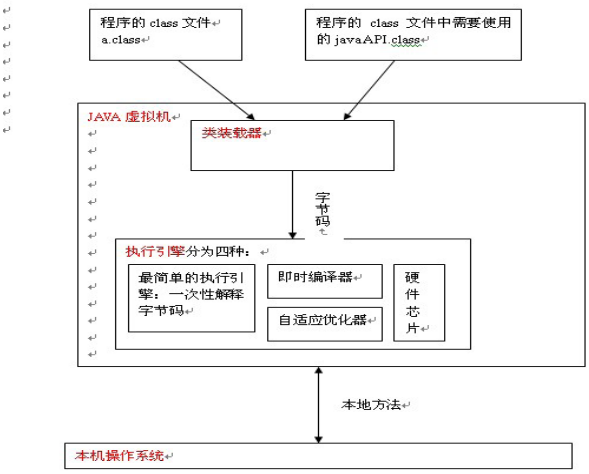
事实上，任何可以使用有效class文件表述的功能性语言，都可以运行在JVM上。动态语言的灵活性，尤其是脚本语言，对于实验性、原型应用程序以及需频繁更新的程序，都具有独特的吸引力。

Java开发人员也可以使用动态语言进行部分编程，然后将这部分代码转换成Java，或继续用Java来开发程序中更为健壮、生命力更强的部分，其他代码则可以通过更加动态的语言来进行开发。动态类型固有的灵活性与JVM的执行效率，合二为一。很明显，这就是它能够吸引动态编程语言创建者以及使用这些语言构建应用程序的开发者的原因。

跨平台性

如果单单说JVM支持动态语言编程就能够吸引更多的开发者那是不可能的，曾有这么一句话“一次编程，到处运行”，这句话说的就是JVM的跨平台性。指即不依赖于操作系统，也不信赖硬件环境。

那么JVM是如何跨平台的呢？下面提供张JVM的结构图



JVM 平台开发语言 一直在流行 II

JVM 跨平台的四种执行引擎部分说明：

1、解释器

简单，一次性解释字节码。易于实现但是执行缓慢。

2、即时编译器

将第一次执行的字节码编译为本地机器代码。编译出的本地机器代码会被缓存，第二次调用的时候可以重用。执行速度快，但是消耗更多内存（与解释器相比）。

3、自适应优化器

虚拟机开始的时候是解释字节码，但是他会监视运行中的程序，并记录下使用最频繁的代码段。程序运行的时候，将最频繁的代码编译成本地代码，其他使用不频繁的代码，继续保持为字节码。

4、硬件芯片：

用本地方法执行字节码。java 通过编译器后生成 class 文件，为字节码，通过虚拟机编译后形成机器码，电脑上只能运行机器码。字节码是可以运行在任何支持 java 虚拟机的硬件平台和操作系统上的二进制文件。

（摘自：bobiy45785 的博客）

混合编程

JVM 平台能够受到开发者的青睐还有一点就是能够进行多语言混合编程，也就是说在 JVM 平台上不仅 Java 一种编程语言可以使用。事出于单一的 Java 开发已经无法满足当前软件复杂的需求。而混合编程也受到了开发者的热捧，至今已成为主流。

JVM 不断的向多语言方向发展，每种语言都可以针对自己擅长的方面更好的解决问题；日趋复杂的软件需求也使得混合编程的应用场景更加的频繁。在不远的未来我们将会看到，我们的项目中，并行进程用 Clojure 编写，展示层使用 Jruby/Rails，中间层用的是 Java 编写。

总 结

多年来，在 JVM 上运行的语言越来越多。而动态语言、跨平台性、混合编程的特性对开发者是非常具有吸引力的。随着 JVM 平台开发的流行，Java 开发者也跟着发生了一些改变，因为他们不仅只限于 Java 一种编程语言。因此，51CTO 还针对 JVM 平台的 9 种编程语言出了个专题，开发者可以进入详细了解有关 JVM 平台的编程语言■

开发小 TIP

在 J2ME 开发中，Jbuilder 是一个很强的工具。而 J2ME 程序使用的内存是越小越好，下面就介绍一下使用 Jbuilder 来优化一下 import 语句。

在写程序时，为了简单，我们经常会使用 import 引入整个包，例如：

import javax.microedition.lcdui.*;

但是我们可能在使用时，只使用了这个包中的一个或者几个类而已。而每个引入在运行时要为包中的每个引用都分配内存（虽然很少），那么有没有办法将对于整个包的引入转换为对于包中类的引入呢？

Jbuilder 提供了这个功能，将对整个包的引入转换为对于程序中使用到的类的引用。操作方法如下：

“ Project ”菜单下面的“ Project Properties ”菜单，然后在“ Formatting ”属性页中找到“ Import ”中的“ Threshold ”，将界面中的“ Always import classes ”选项选中，按“ OK ”关闭设置。

然后，就可以在项目文件列表中选中你需要格式化的文件，在右键菜单中选中“ Format 文件名 ”，就可以优化 import 了。

程序员的一天规划 只用4小时编程

作者 / 佚名

每个人都熟悉这种作息规律：早上 9 点去上班，坐在电脑前面，编一天的程序，下午 5 点下班回家。如今，非常感谢蒂莫西 · 费里斯 (Timothy Ferriss) 的《每周工作 4 小时》，我开始重新思考应该如何工作，如何让自己变成更有效率的程序员。

最近，我把我的从周一到周五的作息规律做了一次较大的调整。很长时间以来，我一直像所有其他程序员那样工作、休息。但就在 2011 年的下半年，我开始了一项试验，想看看究竟什么样的作息时间能让我更有效率。这项实验目前仍在进行中，我并不是像军人那样严格遵守实验规定——例如，我也想会早 20 分钟、或晚 20 分钟起床——但当前制定的作息规律是这样的：

早上 4 :30 到 7 点 :冥想，写作，目标复查，和家人吃早餐

早上 4：30 起床其实并不是你想象的那么难。每个人的个人情况都多少有些不同，但人的身体基本上需要每天 7 到 9 小时的睡眠时间。保证你睡眠充足的一个方法是不用闹钟自然醒。你只需要早点睡觉，你就可以在早上 4：30 醒来。

起床之后，我会马上喝上 16 盎司的水——不是咖啡！我很长时间都不喝咖啡了，而且也不太想喝。事实上，不喝咖啡我感觉会更好。然后我会冲个澡；这样会让我感觉一个清爽的一天的开始。

每天早上我都会冥想 30 分钟。冥想的最佳时间是在日出之前或日出的过程中，也就是西方世界的早 6 点之前。我不打算解释为什么冥想会对你有好处；网上对此已经有了大量的研究。如果你想找一个关于冥想的书，我推荐《Meditation for Dummies》。尽管书名很怪，但它是我读过的最好的一本书。

冥想之后，我会花 30-45 分钟的时间写博客。我一般会写 500 到 800 字。我发现，冥想之后立即动笔，通常会一气呵成，写的很顺利。另外，大脑经过了一夜的休整，状态非常好，在注意力转移到其它事务之前，把大脑里堆积的东西都倒出来清理一下是很有好处的。关于写作的一个技巧：把这段时间用作对大脑的初步清理。不要去做研究、编辑等。这些事情放在以后再做。

然后是 To Do List(待办清单) 时间。我会查看邮件，微博，LinkedIn，等等，以及安排下一步要做是任务。说到任务，我遵循 GTD 做事方法，我用 Omnifocus 软件来管理我的生活。我在 iPad，iPhone 和 MacBook Air(我是苹果的粉丝) 都装了它。不错，作为一个待办事宜管理软件，Omnifocus 是有点贵了，但因为我的整个生活都和它有关，这个价格也值了。我通览一下待办清单，想想每个事情上的预期目标——所有的事情，从今天要做的小事情，到长期的目标。清单中的每个条目都设定有一个目标，如果没有目标，我会删除它。

接着是早餐时间。关于应该什么时候吃早餐、应该吃什么的问题上，已经有大量你可以借鉴的养生指导。我尝试过各种不同的东西。我发现那些富含纤维素，低碳，高蛋白质的东西最

程序员的一天规划 只用 4 小时编程 II

适合我。你试过燕麦片加花生酱吗？好极了！我还喜欢吃一些水果、喝一点茶。还有，我尽量和家人一起吃早餐。有时事情能按照这种愿望执行，但有时不行。我的目标是今年一年都要按照这个执行下去。

早 7 点到 11 点 :4 个小时的编程时间

这是我用来编程的时间。一天 4 小时也许你会觉得少的可怜，可是我却发现，在这 4 小时我做的事情能比大多数人一周干的事情都要多。研究显示，具有固定工作时间表的人比那些随机工作的人更有效率。对于我来说，这个固定时间是早 7 点到 11 点，每天。这段时间我要做的事情就是编程，不做其它的。有几个基本的原则：

首先，关掉所有的通信设施——电话，邮件，聊天工具等。没有让你分心的事。你可以给少数几个人保留一个联系到你的方法，以防有紧急事情发生。那些真有紧急事情需要找你的人自有办法联系到你，我还没有碰到过这样的事。我甚至还教育我的妻子，她通常习惯对所有的请求都立即给予回应，我告诉她要尊重我这 4 个小时的时间。在这段时间里你应该只干一件事。千万不要同时干 5、6 件事情。

第二，中间不要留下休息时间来查看邮件或上网，或干其它类似的事情。原因是：在一个小时里，我可以开发出 x 个功能。如果我要是连续工作 4 小时，我发现我的产量不是 4 倍，而是 8 倍或 16 倍。当你全神贯注的干某项事情时，相信你也经历过这样的体验。这就是我们所说的大脑的 Flow(流) 状态。在以后的博客里我打算多写一些关于 Flow 的文章。

那为什么不把这样的制度应用到整个 8 小时的工作时间里呢？这是因为人的有效率的状态是有限度的。人的大脑跟肌肉一样。你可以在跑步机上持续运动 8 小时吗？就像我们的肌肉，大脑需要时不时的休息。这种限度依据个人的不同而各异。通过尝试和根据犯下的错误，我发现我的极限时间是 4 小时。

还有一点需要提的是，我并没有在 11 点设了闹钟提示。当我感觉大脑有些疲倦，工作效率开始下降时，我就停止工作。有些天我只工作 3 小时，有时我会工作 5 小时；4 小时是平均值。

我在家工作是为了避开打搅。如果你需要在办公室里工作，看看管理部门是否允许你把最有效率的这段时间放在家里工作。你上班途中的折腾会把早上做瑜伽和冥想获得的好处给抵消了。在经过了早高峰的喧闹，还有办公室里的嘈杂，你的神经会变的紧张，冥思带来的效果完全消失。在家工作必定会更有效率的多。

早 11 点到下午 1 点 :健身，午饭，购物

我每天都要健身。John J. Ratey 的书《Spark: The Revolutionary New Science of Exercise and the Brain 》对每天锻炼的好处做了很好的论证，如果你想知道锻炼对大脑功能改进的科学机理，你可以读一下这本书。每天不做相同的运动，甚至不去相同的健身房。我每周在一个瑜伽馆里做 3 次瑜伽，在一个 spinning 馆里上两天的 spinning 课，在健身房里做两天的举重，在健身房里我有个教练。教练帮助了我很多，有人这样督促你会使你具有更大的动力。

我喜欢在健身房运动，因为哪里有额外的服务。你可以一下用掉 5 条毛巾而不担心老婆的抱怨。可以花 30 分钟冲澡而不用担心门外有人叫

喊”你还没完吗？“…在家里你经常会遇到这样的事。

我还养成了一个习惯就是每天购物，通常是在 Whole Foods，一个离我家只有步行距离的超市。为什么每天购物？在很多国家，特别是亚洲，人们每天都去购物，而不是一次买足 2 个星期的东西储存在家里的冰箱或冷柜里。这种方式，你只买了你需要的东西，避免了浪费。很多时候你会发现冰箱里有些不知是何物的东西，怀疑放在那里有半年之久了。我午饭在外面吃，Whole Foods 超市里有不错的沙拉自助。因为我喜欢日本食品，有时我会来一点日本寿司或盒饭。

下午 1 点到 6 点 :学习和交流时间

我尽量会把一些讨论、约会时间凑到一起，这样不必每天都去公司。通常，这些事情包括会议，面试，做报告，指导开发，代码审查等等。这段时间我不做任何的开发，除非有紧急的 bug 或特殊情况需要处理。

我还用很大比例的一部分时间去学习。我花了很多时间去阅读别人的东西，从书本到博格，大多与编程相关的东西。我每天都要学到新的知识。保持这样的学习劲头的最好的方法是对学到的东西做一些笔记，把相关的知识做写标记。像 EverNote 这样的软件很适合做这种事情。同时我会看看市场上有没有其它的产品，最有效率的软件开发者是不用写一行代码就能把问题解决。我不想把别人已经做好的东西再做一遍。所以，在 CodePlex, GitHub 和 Component Source 这些网站上花时间是有好处的。从妻子的医院实习的经历中得到了一点启示。新入职的大夫每天早上看病，到了下午，他们会聚到一起讨论遇到的各种病症，讨论他们是如何应诊的。对于我，我会事后反省一下自己做过的事，我会重新思考一下早上编程遇到的问题，在什么地方遇到了什么 bug，或反省做报告时的表现，或如何主持的会议，等等。我努力从过失中学到教训，避免它们再次出现。

下午 6 点到晚上 8 点 :家庭时间

我妻子是个医生，工作很忙，但我们尽量一起度过这段时间。我们会一起做饭。然后我会和孩子一起做家庭作业。

晚 8 点到 8 :30 :反省和给大脑派活

我晚上不做冥思，而是换成坐下来反省一天的生活。如果你花上 15 分钟做在一个安静的地方反省你的一天，你会吃惊的发现这一天你学到了这样多的东西，获得了这样大的提高。

接下来，我给大脑安排工作。众所周知，当我们睡眠时大脑是在不停工作的。所以，你最好给它安排点工作。对于我，下面这些事情很有效：我把第二天早上要写的东西做了个大纲。就像写便条一样。我发现把它写到纸上比写在电脑里更有效。或者，我会看一些编程 / 算法问题。同样，会纸上描述下来或写出框架。让人惊奇的是，很多时候早上起来你会发现已经有了解决方案！像这样的事情我是在一个整洁的、没有格线的、信封大小的笔记本上写画的。在空白的白纸上写画会让我更有灵感。我在晚 8：30 到 9 点间上床睡觉。越早越好。

这就是我一天的作息时间。当然，当我外出旅行时会有些变化；在途中我会花大量的时间来工作。我还没有告诉你们在周末我都干些什么，我会在以后的文章里讲到这些。■

Java六大必须理解的问题

对于这个系列里的问题，每个学 Java 的人都应该搞懂。当然，如果只是学 Java 玩玩就无所谓了。如果你认为自己已经超越初学者了，却不很懂这些问题，请将你自己重归初学者行列。

对于这个系列里的问题，每个学 Java 的人都应该搞懂。当然，如果只是学 Java 玩玩就无所谓了。如果你认为自己已经超越初学者了，却不很懂这些问题，请将你自己重归初学者行列。内容均来自于 CSDN 的经典老贴。

问题一：我声明了什么！

```
String s = "Hello world!";
```

许多人都做过这样的事情，但是，我们到底声明了什么？回答通常是：一个 String，内容是“Hello world!”。这样模糊的回答通常是概念不清的根源。如果要准确的回答，一半的人大概会回答错误。

这个语句声明的是一个指向对象的引用，名为“s”，可以指向类型为 String 的任何对象，目前指向 "Hello world!" 这个 String 类型的对象。这就是真正发生的事情。我们并没有声明一个 String 对象，我们只是声明了一个只能指向 String 对象的引用变量。所以，如果在刚才那句语句后面，如果再运行一句：

```
String string = s;
```

我们是声明了另外一个只能指向 String 对象的引用，名为 string，并没有第二个对象产生，string 还是指向原来那个对象，也就是，和 s 指向同一个对象。

问题二："==" 和 equals 方法究竟有什么区别？

== 操作符专门用来比较变量的值是否相等。比较好理解的一点是：

```
int a=10;
int b=10;
```

则 a==b 将是 true。

但不好理解的地方是：

```
String a=new String("foo");
String b=new String("foo");
```

则 a==b 将返回 false。

根据前一帖说过，对象变量其实是一个引用，它们的值是指向对象所在的内存地址，而不是对象本身。a 和 b 都使用了 new 操作符，意味着将在内存中产生两个内容为 "foo" 的字符串，既然是“两个”，它们自然位于不同的内存地址。a 和 b 的值其实是两个不同的内存地址的值，所以使用 "==" 操作符，结果会是 false。诚然，a 和 b 所指的对象，它们的内容都是 "foo"，应该是“相等”，但是 == 操作符并不涉及到对象内容的比较。

对象内容的比较，正是 equals 方法做的事。

看一下 Object 对象的 equals 方法是如何实现的：

```
boolean equals(Object o){
return this==o; }
```

Java 六大必须理解的问题 II

Object 对象默认使用了 == 操作符。所以如果你自创的类没有覆盖 equals 方法，那你的类使用 equals 和使用 == 会得到同样的结果。同样也可以看出，Object 的 equals 方法没有达到 equals 方法应该达到的目标：比较两个对象内容是否相等。因为答案应该由类的创建者决定，所以 Object 把这个任务留给了类的创建者。

看一下一个极端的类：

```
Class Monster{
private String content;
... boolean equals(Object another){ return true;}
}
```

我覆盖了 equals 方法。这个实现会导致无论 Monster 实例内容如何，它们之间的比较永远返回 true。

所以当你是用 equals 方法判断对象的内容是否相等，请不要想当然。因为可能你认为相等，而这个类的作者不这样认为，而类的 equals 方法的实现是由他掌握的。如果你需要使用 equals 方法，或者使用任何基于散列码的集合（HashSet,HashMap,HashTable），请察看一下 java doc 以确认这个类的 equals 逻辑是如何实现的。

问题三：String 到底变了没有？

没有。因为 String 被设计成不可变 (immutable) 类，所以它的所有对象都是不可变对象。请看下列代码：

```
String s = "Hello";
s = s + " world!";
```

s 所指向的对象是否改变了呢？从本系列第一篇的结论很容易导出这个结论。我们来看看发生了什么事情。在这段代码中，s 原先指向一个 String 对象，内容是 "Hello"，然后我们对 s 进行了 + 操作，那么 s 所指向的那个对象是否发生了改变呢？答案是没有。这时，s 不指向原来那个对象了，而指向了另一个 String 对象，内容为 "Hello world!"，原来那个对象还存在于内存之中，只是 s 这个引用变量不再指向它了。

通过上面的说明，我们很容易导出另一个结论，如果经常对字符串进行各种各样的修改，或者说，不可预见的修改，那么使用 String 来代表字符串的话会引起很大的内存开销。因为 String 对象建立之后不能再改变，所以对于每一个不同的字符串，都需要一个 String 对象来表示。这时，应该考虑使用 StringBuffer 类，它允许修改，而不是每个不同的字符串都要生成一个新的对象。并且，这两种类的对象转换十分容易。

同时，我们还可以知道，如果要使用内容相同的字符串，不必每次都 new 一个 String。例如我们要在构造器中对一个名叫 s 的 String 引用变量进行初始化，把它设置为初始值，应当这样做：

```
public class Demo {
private String s;
 ...
 public Demo {
s = "Initial Value";
} ... }
```

而非

```
s = new String("Initial Value");
```

本文更详细内容，请查看：

http://developer.51cto.com/art/201102/245676.htm ■

在Eclipse中用Scala语言开发安卓应用

本文将介绍如何在 Eclipse 中用 Scala 语言开发 Android 应用，Scala 是一门现代的多范式编程语言，志在以简练、优雅及类型安全的方式来表达常用编程模式。

1、下载安装 Eclipse Classic 3.7.2

注：下载 JDK 与 Android SDK 的事我就不详述了，有问题可以问我

2、安装 Android ADT Plugin

安装地址：https://dl-ssl.google.com/android/eclipse/

3、安装 Scala IDE

安装地址：http://download.scala-ide.org/releases-29/milestone/site/

4、安装 AndroidProguardScala

安装地址：https://androidproguardscala.s3.amazonaws.com/UpdateSiteForAndroidProguardScala

5、创建 Android Project

6、Add Scala Nature

在 android 项目上点右键，Configure->Add Scala Nature

7、Add AndroidProguardScala Nature

在 android 项目上点右键，Add AndroidProguardScala Nature

8、调整 Java Build Path 顺序（重要）

注意把两个 scala 的 lib，移动到 Android 上方

9、把 Java 的 Activity 类替换为 scala 的 Activity 类

```
package org.noahx.scalaandroid

import android.app.Activity
import android.os.Bundle
import android.widget.Button
import android.widget.TextView
import android.view.View

class ScalaAndroidActivity extends Activity {
  override protected def onCreate(savedInstanceState: Bundle) = {
    super.onCreate(savedInstanceState)
    setContentView(R.layout.main)
    val text = findViewById(R.id.text1).asInstanceOf[TextView]
    val button = findViewById(R.id.button1).asInstanceOf[Button]
    button.setOnClickListener(new View.OnClickListener() {
      def onClick(v: View) = {
        text.setText("hello scala!!!")
      }
    })
  }
}
```

注：修改完 layout，把 project clean 一下。

运行效果地址：

http://developer.51cto.com/art/201206/341131.htm ■

北上广之的IT技术人生

编者按

北上广深，是很多人心目中的一线城市。这里有流光溢彩的城市风景，也有无数的成功机会。大家来到这里，就为了成为金字塔顶端那10%精英。

据51CTO之前对WEB开发技术人员的调查，详情点击这里。一大部分Web开发人员的工资集中在3000到6000这个区间。而根据笔者对于老家一位高中同学的调查，他在大专毕业后在当地一家旅游行业私企中做网站维护，每月工资在2000元左右。

附某三线城市IT人员工资表（该市GDP总量位居某省前三，人均GDP刚过3000美元）

电脑城员工1000至1500元；平面设计（普通1500元，老手级别2800元）；

普通网管1500以下；网站编辑1500到2000（职位仅限于政府机关）；高级工程师3000以上。

从上面案例可以看出，在三线城市的WEB开发人员和一线城市的同行，工资平均差距在3000元人民币以内。在抵消掉房租和其他生活成本差距后，二者实际工资的量应该不分伯仲。在北上广深的房价如火箭般蹿升之后，打破了很多程序员在一线城市安家的梦想。由此在2010年，爆发了第一轮逃离北上广的浪潮。

机遇与挑战共存

有的开发人员回到老家后，反映自己在家里更不适应工作环境。什么都需要人脉，需要有关系。即使靠家里进入了体制内的生活，也不太适应那种散漫、墨守成规的生活。这样整天混日子的生活方式，又逼迫很多人逃回北上广。毕竟，人是闲不住的动物。

但二线城市的IT从业环境并不是一尘不变的。在51CTO记者对来自湖南的开发者小庄进行采访时，发现类似长沙这样的城市，软件开发的需求正在膨胀。以往在很多北京的软件公司看来还不太容易实现的敏捷开发原理，在小庄的公司当中已经有所应用。而为了更好的了解敏捷开发，他们还会不远千里来到北京、上海这样的城市取经。

上面就是两种不同的遭遇。我们该如何取舍？

创业好玩么？

在二线城市，会有不少程序员带着在北上广的积蓄进行创业。笔者的一位朋友，在北京打拼4年，带着一点积蓄回到老家吉林松原。

由于之前也是在网站做技术和产品方面的工作，所以他毫不犹豫的选择建网站。在二线城市，专门针对当地的地方网站并不多，看上去竞争对手就只有那么几个而已。不过，问题就出现在这里。

首先，由于对当地市场容量估计不足。网站上线后，才发现浏览量明显不足。尽管在百度或Google的排名中能占据前三位，但整个城市的人口就那么多，市场范围狭窄。直到一年之后，整个网站似乎要山穷水尽之时，才

北上广之外的 IT 技术人生 II

拉到了当地国美的单子。整个网站才勉勉强强进入正轨。

其次，技术力量不足。由于城市本身 IT 行业就不是很强大，所以经常会出现网站改版，但美工和 PHP 程序员很难同时给力的现象。很多技术实现需要创业者自己来完成，势必会影响到广告招商等业务的发展。

最后是政府方面的压力。比如当地出现某重大事件，你的网站是否报道？报道了肯定流量有保证，但事后的风险是否可控？

上面三个问题是笔者朋友自己创业中碰到的主要问题，其实还会包括自己资金链、竞争对手打压和家人压力方面的问题。我想其中任何一个，都会成为创业失败的原因。

在 51CTO 记者即将结束本文的时候，正好参与了百度开发者大会 · 成都站的报道工作，在会场也随机采访了一些开发人员。他们普遍都表示二三线城市的开发者，在技术培训和经验再提升方面还是略微落后于北上广这样的大城市。毕竟很多开发方面的技术专家和新的技术信息都集中在北上广，而这些专家想去到成都这样的城市，还是需要更多的支持才能做到。成都本地的程序员，又不可能都跑到北京去参会，这样就会造成一种信息沟通的不畅。虽然互联网技术，特别是视频连线技术可以弥补这样的距离鸿沟，但面对面的交流目前还无法被取代。

所以 51CTO 记者在这里还是建议有技术理想的技术人员，可以先在北上广捶打几年。在经验和眼光都成熟之后，再回到二三线城市进行创业。比如最近成都高新区就表示能为所有达到正常要求的 IT 创业企业，免费提供办公场地，以低廉的价格提供开发环境、网络支持和硬件支持。这样的扶持政策也更有利于普通开发者在创业初期避免不必要的失败而胎死腹中。

51CTO 策划专题《大数据“三重门”》

大数据时代来临，Hadoop、Mapreduce 成为炙手可热的新技术词汇。从技术的角度来看 51CTO 认为可以将大数据技术分为三大层次：应用层、数据分析和基础架构层。希望本专题能让广大技术人员更好的了解大数据技术。

应用层：

数据分析：

基础架构层：

专题地址：
http://developer.51cto.com/exp/bigdata/

编者按

2012 年 5 月 23 日夜，硕大的空客 A330 客机在成都上空不停颠簸，看来今天天气预报并没有骗人。当 51CTO 记者走出机场坐上去酒店的出租车，蓉城开始下起倾盆大雨。欣赏着大雨中的夜色，让人不禁想象“明天我们将要见到的成都程序员们，他们会是怎样的一种状态？”

程序员就应该生于忧患死于安乐？

在北京召开的形形色色技术大会中，我们总能接触到一些来自四面八方的开发者。他们大多是公司安排出行，毫无经济压力。但我们想，肯定有更多的开发者必定没有这样的好运，只能在媒体的报道和专家的博客中找到更多有用的信息。

如同在这次百度开发者大会成都站的现场，51CTO 采访了一位姓杨的女程序员。她所在的公司有 60 人的规模，技术部大约 10 个人左右。这样规模的公司，恐怕是很难安排自己所有的技术外出参会提高的。杨女士一开始甚至误以为本次百度的活动是由四川当地分公司举办而非百度北京总部。有部分成都参会者来到这次活动，更多的是希望了解基础架构方面的分享。这或许是他们平常会遇到的问题，需要从讲师的技术分享中获得自己想要的东西。

北京比成都好在哪里？

中国 IT 产业如同其他产业一样，都是富集于北上广深这样的一线城市。特别是软件开发这样的资本密集型和技术密集型产业，更需要大量的资金和高水平人力资源的支持。北京之所以软件业发达，与清华、北航等众多重点高校聚集有关。即使中国高等教育何等薄弱，这些学校已经聚集了中国顶尖的学生，他们成为北上广发展 IT 产业的基石。

北京能给程序员们开出不错的薪水，至少比在家乡要高的多。即使我们每天挤在拥挤的 300 路公交车和一号线地铁里，也还是要坚持自己的梦想，这种苦难真的是“生于忧患？”

为了给各位北京的开发者做不同的体验，51CTO 记者特意在周四的早晨 8 点挤上了成都的 118 路公交车。即使天下着小雨，车里依旧有很多空间。倘若这样的情况在北京，恐怕我们早就前胸贴后背成了沙丁鱼罐头了。即使到了成都春熙路这样的市中心，记者发现临近十点还有很多商店都没有开门，包括国美苏宁这样的大型家电卖场。颇似东南亚的马来西亚一般慵懒。

成都的出租车司机告诉记者，成都人晚上九点才算夜生活开始。而我们北京的程序员们恐怕九点就得往回赶，要不没了公交和地铁，就只能黑车咯。相比之下，成都确实安逸，安逸得让人觉得有种会“死于安乐”的错觉。

成都能给想来的开发者什么？

在本次百度开发者大会成都站上，成都高新区管委会副主任傅学坤同志就表示。成都高新区非常欢迎软件企业前来落户，只要项目需求合理，将会提供免费的办公场地。同时，高新区还会对软件企业在开发环境和网络环境等方面，提供类似补贴的援助，帮助企业做大做强。对于需要

程序员就应该生于忧患死于安乐？II

接触天使投资的软件开发企业，高新区也将从中牵线搭桥。

在地产经济越走越窄的今天，全国各地的政府都需要考虑“后房地产时代”的发展问题。有的省份搞动漫文化产业，有的省份搞旅游产业，而更多的企业想到了建设高新区或者云计算中心。希望通过搭上高新技术的火箭，减少 GDP 对房地产的过度依赖。

百度能给成都开发者带来什么？

云计算的优点已经被媒体炒得天翻地覆，可以企业成本、可以减小人员需求、可以更快的响应需求等等。而百度移动 . 云总经理李明远表示：“开放百度云计算平台、与开发者共享百度在云计算技术、数据、流量和变现方面的优势，携手合作伙伴共建平等、开放、共赢的生态系统，让人们最平等便捷地获取信息，找到所求。”

百度与成都高新区达成合作，建立“百度云应用开发区”，此举开辟了互联网公司与传统软件园的新型合作模式，共同扶持开发者，打造应用为王的时代，建立良性生态圈。将通过全方位开放和全流程服务，谋求与当地园区深度合作，以成就每一个开发者的梦想。

而百度移动 . 云事业部架构师郭杏荣的分享首先从百度存储技术的演化谈起，地域集群化，区域 CDN。云存储的好处就是企业能专注在业务领域，而不必为存储的技术细节担忧。开发者和用户使用云存储将更加迅捷。百度将帮助开发者更快的开发和迭代，这里郭杏荣举了百度贴吧快速开发产品的例子。

除此之外，百度开发者中心中的百度云开发者服务首先上线对 PHP，Java 和 Python 的语言支持，相信 Web 开发者会首先受益。地图 API 是各位开发者很熟悉的接口，未来将会用到翻译 API、用户 API、输入法 API 等等。这些将会更好的支持开发者的工作。

我们能在成都做什么？

逃离北上广，去巴蜀之地无疑是个不错的选择。各位北京的程序员，或许我们不用每天去挤公交挤地铁，吃着味同嚼蜡的盖饭，看着皮肤粗糙的北方妹子。四川吃多久都不腻的美食和皮肤吹弹可破的川妹子，对于我们似乎更有诱惑力。

工作环境和压力都相比北京要安逸许多，加上软件开发行业在成都起步比北京要慢。所以在北京可能只是一个宅男程序员的你，来成都或许能开辟属于自己的一片天地。倘若错过了这一轮机会，当类似成都高新区这样的地方招到了足够的企业时，还有没有上面提到的优惠措施，就很难保证了。

所以作为程序员的你，厌倦了北京 PM 2.5 超标的空气，去成都换换脑子吧。■

开发小 TIP

构成一个优秀程序员的一些特征：

以人为本

对学习有很大的胃口

掌握问题规律的本领

有一点神经质

执着

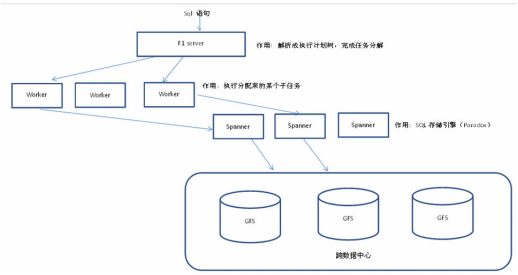
编者按

F1 是一个从头开始建立的新数据库，其设计指标符合 Google 所需的规模，与此同时不会影响 RDBMS 的功能。F1 的关键特性包括：高扩展性（自动分片存储）；可用性和一致性（同步复制）；High commit latency: Can be hidden（分层架构、协议缓冲列类型、高效客户端代码）

Google业务核心从MySQL迁移至F1

Google 广告系统原先采用 MySQL 作为后端数据库，现在部分系统迁移到 F1 上面，占比数据不详知。但是一项广告核心业务已经迁移到 F1。F1 的特点并不是像赛车一样的快，为了支持下面这些特性，F1 付出了对响应时间的要求。不过响应时间还是可以为 OLTP 业务所接受。

1. 结构化数据存储；2. 并行 SQL 执行引擎；3. 通用的事务支持；4. 触发器；5. 支持索引；6. 动态扩展；7. 跨机房同步。



附 F1 数据处理流程图

对比 MySQL 较低的延迟写入，F1 的强一致性和存储系统更加具有吸引力。在成功迁移到 F1 后，Google 的广告业务核心将丰富 customerfacing 应用套件，使整个系统没有停机时间，如何重组架构和应用和在最大程度上隐藏来自外部用户增加的延迟，将在今后的文章中被提到。 F1 的分散性，还允许它轻松扩展，并支持比传统的 RDBMS 更高的批处理工作负载吞吐量。

Google 研究院推出被称为 F1 的新型数据库，F1 作为一种混合型数据库融合了 BigTable 的高扩展性和 SQL 数据库的可用性和功能性。F1 数据库作者共有 12 人，清单如下：Jeff Shute、Mircea Oancea、Stephan Ellner、Ben Handy、Eric Rollins、Bart Samwel、Radek Vingralek、Chad Whipkey、Xin Chen、Beat Jegerlehner、Kyle Littleeld、Phoenix Tong。

F1 的底层文件系统为 BigTable 的继任产物 Megastore，其属性主要包括：全局分布式；同步跨数据中心复制；可视分片和数据移动；常规事务；多次读取单原子写入；本地或跨机器以及快照读取。

架构：Sharded Spanner 服务器；数据存在 GFS 和内存之中；Stateless F1 服务器；工作池提供查询执行

特点：关系模式；扩展到层次和丰富的数据类型；无阻塞的架构变更；一致性指标；并行读取 SQL 或 MapReduce。

F1 当前面临的挑战主要有：并行查询执行、故障恢复、隔离、优化、迁移应用时要求不宕机最后等等。

Google 从 MySQL 迁移到 F1 给其带来了更高的扩展性、更高的可用性、等效一致性得以保证同时兼顾强大的 SQL 查询。总之，Google 的此次数据库迁移在保证数据库规模的同时并没有失去任何的数据库功能。■

大数据技术更需青年力量

大数据技术虽然比数据仓库之父的 Bill Inmon 早在 20 世纪 90 年代就经常将 Big Data 挂在嘴边了，但真正投入实际应用却是最近几年的事情。这样一个正在发展中的技术，更需要青年人的力量

2012 年 6 月 6 日，记者有幸在北京中关村微软亚太研发集团大厦内见到了这些出类拔萃的参赛者和他们的作品。根据参赛要求，他们需要在规定时间内，完成对相应数据量的处理和分析。整个过程只限定大数据可视化处理的大方向，其他细节不做更多限制。

活动当天，记者观看了几位参赛者的大数据处理 DEMO，发现不少同学的集中点是放在大数据信息的图形化展示上，或者对数据的初步筛选。虽然在后期综合分析处理的思路上有所欠缺，但已初见大数据处理的雏形。

IEEE 标准协会董事 Ted Olsen 先生在与 51CTO 记者同时观看一位参赛者的 DEMO 时，对于大数据方面的进步表现得非常激动。Ted 先生说：“我年轻时做过 Coder，在我写代码的那个年代，需要将不同的代码按照不同的颜色进行分类，数据也是如此。现在，即使是海量数据都已经可以做到瞬间处理完成。大数据技术在类似能源、交通领域的应用将潜力巨大。”

当 51CTO 记者与上海交通大学的黄偲进行交流时，发现他的大数据可视化处理 Demo 有其独特之处。首先，黄同学的 Demo 除了有简单的图标分析外，还有不同用户的相似度分析。比如 B75 号用户喜好的论文作者，与 B156 号用户的喜好类似。那么在 B75 号用户的界面，就能看到一些跟他喜好相似的用户提示。这有点类似 SNS 网站中的好友推荐功能。尽管这个功能不是什么新鲜事物，但能够想到并设计出权值计算公式来完成，确实有其与众不同的地方。

其次，在 Demo 中，黄同学还应用了学术圈的概念，论文作者最终会形成一个类似集群的组织。作者之间会有自己的联系线，这些线产生的原因是他们都在研究相似的科学命题。最终在黄同学的学术圈逻辑图上，我们能看到一个学术圈总会有一个“核心”人物，也就是我们常识中的大牛。他的研究与其他人都有很强的联系。而不同的学术圈，会通过一到两个跨学术领域的个人联系起来。这样的数据分析，有点大数据最终数据挖掘的味道了。不出所料，黄同学摘得大赛桂冠，获得了赴美参加电气电子工程师学会（IEEE）大会和前往微软总部参观的机会。

类似黄同学这样具备很强发散思维和奇思妙想的青年人，会成为未来大数据领域新动力。大赛评委会主席邹欣认为，此次参赛选手技术能力强，功底深厚，思维灵活，能将各学科知识融会贯通，在软件开发方面潜力巨大。他特别指出：“希望未来的参赛选手能更具有冒险精神，勇于挑战难度较高竞赛题目。现代软件开发已经进入团队协作时代，同学们需要增强自己学习和理解已有程序代码能力，这将有助于他们更快成长。”■

揭秘Facebook官方底层C++函数Folly

Facebook 近日公布了其官方底层 C++ 函数 Folly，Folly（该缩略语表示 Facebook 开源代码库）其实是 C++11 组件库，这些组件在设计时着眼于实用性和高效率。

Folly 与 Boost、当然还有 std 等组件库的关系是互为补充，而不是彼此竞争。实际上，只有当我们需要的东西既没有，也无法满足所需的性能要求时，我们才开始定义自己的组件。

性能问题贯穿着 Folly 的大部分，有时导致比较具有特质性的设计（比如 PackedSyncPtr.h 和 SmallLocks.h）。整体上确保良好的性能是所有 Folly 的统一主题。

逻辑设计

Folly 是一组相对独立的组件的集合体，有些组件就是几个符号这么简单。内部依赖方面没有限制，这意味着某个特定的 folly 模块可以使用其他任何的 folly 组件。

所有符号都在顶层的命名空间 folly 中加以定义，当然除了宏。宏名称是 ALL_UPPERCASE。命名空间 folly 定义了其他的内部命名空间，比如 internal 或 detail。用户代码应该不依赖那些命名空间中的符号。

物理设计

在顶层，Folly 采用经典的“结巴”（stuttering）方案 folly/folly，这也是 Boost 及其他组件库所采用的。第一个目录充当库的安装根目录（可能是以 folly-1.0/ 这样的形式）；第二个目录是添加文件时用来辨别组件库，比如 #include "folly/FBString.h"。

目录结构是扁平的（模仿命名空间结构），也就是说我们没有复杂的目录层次结构（这个情况在将来的版本中可能会有变化）。子目录 experimental 含有在 folly 里面使用的文件，可能用在 Facebook 端，但是被认为不够稳定，无法在客户端使用。你的代码不该使用 folly/experimental 中的文件，以免你在更新 Folly 时，出现问题。

folly/folly/test 子目录包括了面向所有组件的单元测试，通常名为 ComponentXyzTest.cpp，面向每个 ComponentXyz.*。folly/folly/docs 目录含有说明文档。

兼容性

目前，folly 已在 64 位安装版 Fedora 17、Ubuntu 12.04 和 Debian wheezy 的 gcc 4.6 上进行了测试。它不用改动，就可以在其他 64 位 Linux 平台上运行。

组件

下面按字母顺序介绍了一系列 Folly 组件，另外附有每个组件的简短描述。

Arena.h，ThreadCachedArena.h

内存分配的简单地方：多次内存分配同时被释放。使用线程版本。

AtomicHashMap.h，AtomicHashArray.h

揭秘 Facebook 官方底层 C++ 函数 Folly II

高性能原子哈希图，采用几乎无锁的操作。

Benchmark.h

用于代码基准测试的小型框架。客户端代码注册基准测试，可选情况下使用一个变量来规定基准测试的范围（迭代和工作集大小等）。框架运行基准测试（受制于命令行标记），生成带计时信息的格式化输出。

Bits.h

各种位处理实用组件，针对速度而优化。

Bits.h

位变换函数，使用统一接口包装 ffsl(l) 图元。

ConcurrentSkipList.h

实现了用证实正确的可扩展并发跳跃表（Provably Correct Scalable Concurrent Skip List）描述的结构，这种跳跃表由 Herlihy 及其他人共同开发。

Conv.h

各种数据转换例程（尤其是 to 和 from 字符串），针对速度和安全进行了优化。

DiscriminatedPtr.h

类似 boost::variant，但完全局限于指针。使用指针中最高位、未使用的 16 位作为鉴别器。所以 sizeof(DiscriminatedPtr<int, string, Widget>) == sizeof(void*)。

dynamic.h

动态类型对象，创建时关注 JSON 对象。

Endian.h

Endian 转换图元。

Escape.h

以 C 方式转义字符串。

eventfd.h

针对 eventfd 系统调用的包装器。

FBString.h

嵌入式实现 std::string，进行了诸多优化。

FBVector.h

基本嵌入式实现 std::vector，进行诸多优化。

Foreach.h

伪语句（作为宏语句来实现），用于迭代。

Format.h

Python 式样的格式化实用组件。

GroupVarint.h

针对 32 位值的 Group Varint 编码。

Hash.h

各种流行的哈希函数实现。

Histogram.h

一个简单的类，用于收集直方图数据。

IntrusiveList.h

方便类型定义，用于使用 boost::intrusive_list。

json.h

JSON 序列化器和反序列化器。使用 dynamic.h。

Likely.h

针对 __builtin_expect 的包装器。

Malloc.h

内存分配助手，尤其是使用 jemalloc 时。

MapUtil.h

一种助手，用于查找联合容器中的项目（比如 std::map 和 std::unordered_map）。■

原文未完，请参考 http://developer.51cto.com/art/201206/340607.htm

几种不同的技术面试过程

在大学里面试社团大一新生，到加入百度后帮公司面试候选人，我觉得我对面试这件事一直不得要领。今天就来谈谈理想的技术面试过程。

百度提供面试培训，也允许参考或使用题库，但我还是觉得不知道如何判断给不给一名候选人通过我这关。偶尔我会遇到非常优秀的实习生候选人，我能十分确定我要给他过，甚至想方设法确保他能来。其它时候，我觉得我的判断随机性太大，或许还不如一枚硬币做得好。

在百度做二面的时候，我往往会问一些组合问题，就是候选人需要有扎实的基础加上一定的解题能力才能做出来的。我假设一面的面试官已经问过基础问题，所以我不会再问基础问题。结果通常是，候选人的基础不够扎实，会作出一些错误的假设，甚至面对组合问题就无从下手，不知道如何分解为更小的问题然后再一步一步来解决。我不知道是否应该期望候选人全部答对，但答对小部分的状况让我无从判断。

为此我开始跟其它人讨论面试经验。Acumon 说应该针对候选人说他擅长的领域来提问，而且使用开放性问题以便了解候选人的思考方式，但我发现我遇见的大多数候选人都不清楚自己擅长什么，或者是他们自认为的强项无法达到我的预期。后来在上海跟 Winter 和 Hax 聊天时发现一个可怕的现实：大多数前来应聘的前端工程师都无法回答「position 属性取值都有哪些」以及「display 属性取值都有哪些」。随后我尝试在我的面试中先问这两个问题，发现确实有些人回答不出来取值，更多人则是无法准备描述常见取值的作用。（我甚至不把 inline-block 和 fixed 归类到常见取值里面。）遇到如此基础的问题都回答不出来的候选人，我通常会跟他告诉他正确答案，再问一些基础问题然后给他一些学习建议，最后很礼貌地送他走。

状况在我到达豌豆荚后有所改善。不是来应聘的人都非常优秀，而是我开始有感觉了。关键原因我觉得是我们不着急招人。现在我们还能应付手头上的工作，也没有快速扩张的需求，所以我们只有在遇到最合适的人选时才邀请他加入。我相信我前面的两位面试官做得足够好，然后我可以放胆地去考量「懂不懂」之外的其它方面。准确来说，这甚至不是一种考量，我只想知道我跟这个人一起工作是否会很愉快。我会以工作上遇到问题时同事跟同事间讨论的方式去跟他进行讨论，有可能是工作上实际遇到的问题，也有可能是刻意设计出来的有趣问题。（我觉得在一个充满活力的工作环境中，同事之间互相找些稀奇古怪的问题来讨论是很正常的，大家也会享受这类挑战。）如果我感觉跟他讨论问题是很愉快的过程，他能够提出有趣的想法，甚至能告诉我一些原本我不知道的事情，我肯定会给他很正面面试评价。

几种不同的技术面试过程 II

作为面试者

换个角度来说，如果你作为面试者发现自己在面试的过程中能够进入这种状态，感觉如同跟同事讨论问题一样放松，那么你应该对面试结果充满信心。至少根据我个人的经验来说，感觉如同轻松愉快讨论的面试我都能得到面试官不错的评价。我明白要做到这一点很不容易，很多人在面试时都会很紧张，甚至会假设面试官一定会用各种难题来考自己，这种心态其实会把自己放在不利的位置上。我过去的经验是，我越不在乎的面试，往往我越能放松地跟面试官随便聊，结果反而越好。我很在乎的面试，反而有时候会高估面试官问题的难度，结果简单的问题没给出简单的答案，最后得到的评价反而不好。

在百度大厦的时候，我喜欢穿越二层平台，因为那里有很多小圆桌，也有很多人会选择在那里进行讨论或者面试。如果你对肢体语言有点最基础的了解，或者你就是在这方面有感觉，你会发现在那里很容易看得出哪些桌是同事间的讨论，哪些桌是在面试，以及谁是候选人。同事之间的讨论，往往大家都会很放松；面试的话，面试官也会很放松，但面试者通常处于比较紧张的状态，身体会略为前倾，就算不需要写字也会把手放在桌子上，用于支撑上半身的重量。对于有经验或者有感觉的面试官来说，这种肢体语言会传递一种不好的信号，那就是你太想要这份工作了，就算更深入沟通后可能你会发现这份工作不适合你，但你还是会追求这个机会。

类似的场景也会出现在异性之间，下次去餐厅吃饭时你可以观察一下隔壁桌的异性交互。就算你完全不知道谈话内容，你也可以尝试从面部表情和肢体语言去判断谁在追谁。为什么有时候你追的人各种虐待你，还有充足的信心你不会放弃？因为他们看得出你把自己摆在什么位置上。在统计学的角度来说，我觉得女人在这方面比男人有优势，因为女人看的电视剧都不是白看的，她们在逛街吃饭时还无时无刻地在观察身边发生的各种剧情。就如同资深前端工程师可以只看 CSS 片段就猜出遇到了什么浏览器 bug 以及作者尝试如何解决一样，有经验的女人（又或者是面试官）要看明白你的立场太容易了。

因此，首先你要把面试看作一个平等的双向选择过程，不仅仅公司在选择你，你也在选择公司。如果面试官问的问题显得他很没有品味，或者是 HR 的某些安排让你觉得这家公司的文化很有问题，你随时可以提出说你没兴趣聊下去。这时候你反而可以比较好的发挥出来你真实的实力。对于你真的不知道的问题，你就直接说出哪部分你不知道或者不确定。同事间讨论也会遇到你不懂的问题，你平时是怎么处理的在面试时就怎么处理，面试官不会因为你直率地表达不懂某个点而鄙视你，他应该帮你把这个点绕过去然后继续。

我觉得最好的面试建议就是 Changhao 跟我说的那句，「Be yourself」。（其实这也是很好的追女建议。）

■

原文未完，请参考：

http://developer.51cto.com/art/201206/340805.htm

API正迅速成为Web应用程序粘合

API 使得那些应用程序与设备无关：无论设备是智能手机、平板电脑、个人电脑、数字录像机、自助服务终端、车载计算机、游戏机还是其他平台，都能访问。

作者 /Musser 译 :建苗

据在一年一度的 Glue Conference 上发表主题演讲的一位人士声称，应用编程接口 (API) 正在迅速成为 Web 的应用程序粘合剂，每天数十亿次的调用让一些公司每年赚得钵满盆满。

据 API 聚合网站 ProgrammableWeb 的创始人 John Musser 声称，谷歌、Facebook、Netflix 和电子港湾等在线服务商每天在处理数十亿次的应用编程接口 (API) 调用，一些公司每年通过服务 API 接口而获得的收入高达数十亿美元。

Musser 近日在科罗拉多州布鲁姆菲尔德举行的年度 Glue Conference 上发表了演讲。

他重点介绍了他公司收集的急剧增加的统计数字，并阐述了开放 API 市场的十大热门趋势，包括增长率、风险投资、协议和商业模式。Programmable Web 维护着由众多开放 API 组成的一个庞大数据库。

他说：“API 是我们在将来编写软件的工具。我们将来会用 API 将代码粘合起来。”

Musser 介绍了他所认为的十大 API 趋势

排名不分次序：

1. 风险资金投入到 API 领域
2. 增长率
3. REST
4. JSON
5. API 调用亿万次公司和万亿次公司
6. API 成为一种产品
7. 编程马拉松 (Hackathon)
8. API 商业模式
9. 将 API 变成收入
10. 无形的混合应用程序

由于用户们需要借助任何设备从任何地方来访问应用程序，对在线服务商和企业来说，API 迅速变得必不可少。这种需求促进了 API 迅猛发展。

API 是一组函数，让计算机程序可以相互交流、共享数据。

Programmable Web 的目录中如今列有 6000 个开放 API。

而仅仅三个月前，这个数量还只有 5000。相比之下，这个目录的 API 数量首次突破 1000 大关用了整整八年的时间。那些数字不包括主要用来支持移动应用程序的无数私有 API。

在 ProgrammableWeb 目录中所列的 359 个企业级 API 当中，大概近 15% 是在过去三个月添加的。

企业级 API 和消费级 API 之间的区别主要在于，企业级 API 通常处理更敏感的数据和交易事务。

此外，企业需要使用 OAuth 等协议，管理和保护对那些 API 的访问。

API 正迅速成为 Web 应用程序粘合 II

Musser 特别指出，拿推特来说，API 调用“亿万次公司”每天处理的调用从 2010 年的 30 亿次增加到现在的 130 亿次调用。

Netflix 在本月每天处理的调用是 14 亿次，Klout 也有 10 亿次。在 2012 年的头三个月，电子港湾每天处理的调用是 10 亿次。

而这个数字会在不久的将来会更庞大。他特别指出，亚马逊网络服务公司 (AWS) 的简单存储服务 (S3) 中对象数量下个月会达到 1 万亿个之多。

Musser 指出，Expedia 的联盟网络通过 API 每年获得的收入多达 20 亿美元。Musser 引用 Expedia 高管的话说，他们开展的业务当中有 90% 是通过 API 来实现的。

如今开发人员要求对应用程序中最有用的部分实现可编程访问。而最终用户也在做同样的事，只是他们没有意识到罢了：

他们在大量使用基于推特或基于 Facebook 的应用程序时，或者企业用户在使用基于 SalesForce.com 的应用程序时，就在进行这种访问。Salesforce.com 的流量当中一半以上来自 API。

而 API 使得那些应用程序与设备无关：无论您的设备是智能手机、平板电脑、个人电脑、数字录像机、自助服务终端、车载计算机、游戏机还是其他平台，都能访问。

Musser 表示，许多公司直接用钱来刺激用户在其 API 上从事开发。

他说：“一旦你有了 API，就会考虑如何让别人在上面从事开发。”

他提到了 Twilio 和 Box 这些公司用钱来吸引开发人员在其 API 上进行开发：

以免费试用或经济奖励作为手段，鼓励他们在其 API 上从事开发。

他表示，代表性状态传输 (REST) 和 JavaScript 对象标注 (JSON) 是 API 的两种主要协议。

他还表示，社交型 API 方面的协议 95% 是 REST。JSON 用作 60% 的 REST API 的数据格式。

他说：“JSON 是趋势，但是大多数人没有料到会出现这种情况。今年，近三分之一的 API 采用 JSON 协议。”

他还指出，编程马拉松 (Hackathon) 是把广大开发人员团结在 API 周围的一种时下很流行方法。

2012 年第一季度举办了 160 场编程马拉松活动，去年从编程马拉松可以领到的最高奖金数额是 10 万美金。

Musser 表示，Twilio 和 Stripe(在线支付业) 等公司认为自己的 API 是面向开发人员的一种产品；而 SupermarketAPI(杂货零售业) 等公司将 API 当作一个品牌来使用。

此外 Musser 表示如今出现了多种商业模式

比如亚马逊网络服务公司的按需支付模式和谷歌 AdWords 的基于设备的模式。

他说：

“最大的趋势也许是间接模式，”这包括一次性注册 (Jigsaw)、内容整合 (《纽约时报》以及移动设备等内部使用 (Netflix)。

看看九种编程语言发明者是怎么说的

简介

果你写的只是一个简单的排序，用Python来完成的话，那这会成为系统的瓶颈。这里最好要用高效的语言来取代之，比如C和C++。

从Node.js到C++，看他们的发明者是如何评价他们的语言的未来。

Ryan Dahl :Node.js

问：它的主要优势是什么？

Dahl：Node与其他的语言有一点明显的区别，就是处理I/O。所以它永远不允许用户锁上程序。它要求用户不断的处理新事物，因此它很适用于网络编程。在你的服务器上要与很多人打交道，你必须处理链接。Node鼓励人们用非阻塞的模式。由于这个特性，你会发现Node在开发服务器上比传统编程语言更加方便。

Guido van Rossum :Python

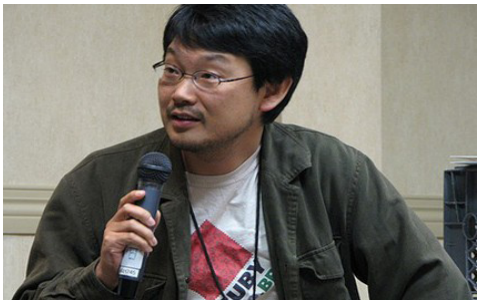
问：为何有人批评Python太慢？

Van Rossum：有些时候，你要完成的系统某一小部分，而这一部分却花费你几乎所有的时间。如果你写的只是一个简单的排序，用Python来完成的话，那这会成为系统的瓶颈。这里最好要用高效的语言来取代之，比如C和C++。

Yukihiro Matsumoto :Ruby

问：你的语言不断发展的目标是什么？

Matsumoto：我的目标就是让程序员更开心。目前Web程序员已经很开心了，但不够，我希望可以帮助更多的程序员。如嵌入式的，还有高性能程序的程序员。我目前正在开发Ruby语言的子集，主要应用于移动设备，打算明年年初将它公开。



Dmitry Jemerov :Kotlin

问：我们为什么要用Kotlin，我们已经有Groovy或者JRuby

Jemerov：Grooby和JRuby是动态类型语言，如果你在开发中小型web程序，这2者确实是很不错的选择。但如果你要开发更复杂，更高性，高频率交互的程序的时候，Kotlin这个静态类型语言会更适合你。

看看九种编程语言的发明者是怎么说的 II

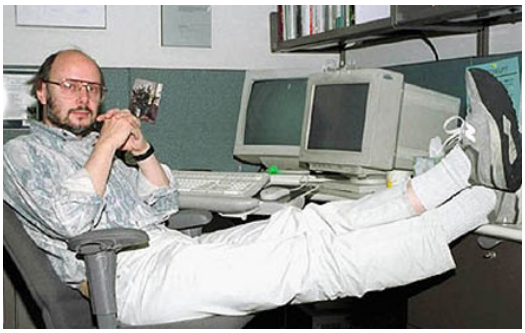
Martin Odersky :Scala

问：你当初发明 Scala 的目的是什么？

Odersky：我很好奇，是否可以将函数式和面向对象编程结合成一个包，给开发人员提供一个强大的语言，让人感觉他有很强的互补性。函数式编程感觉非常清爽，而且实用，它可以轻易构造简单的元素。而面向对象对于大型系统来说有很好的故事背景。总之我觉得把这 2 者相结合，是 Scala 的成功原因之一。

Bjarne Stroustrup :C++

问：什么时候一个程序员应该选择本地化语言，什么时候选择基于虚拟机的语言？



Stroustrup：C++ 在基础架构方面有无与伦比的优势。换句话说，在性能，可靠性，资源，复杂性方面都有严格的要求。例如，你不会用 javascript 写 javascript 引擎，你也不会用 C++ 去写简单的 web 应用。你会用 C++ 构造谷歌，亚马逊，Facebook 的基础，但不是顶层。C++ 在服务器市场和移动设备上有很强的优势。

Lars Bak :Dart

问：据我所知，Dart 实际上是编译为 JavaScript，那为什么不直接用 JavaScript 呢？

Bak：因为我们有…Dart 虚拟机，可以更快的运行和启动。JavaScript 给我的印象是大的应用中需要很长的时间来启动。如果你拥有了 Dart 虚拟机，那你可以将程序启动提速 10 倍。如今我们看到 Web 应用变得越来越大，越来越广泛，启动的速度是很重要的。

Stefan Karpinski :Julia

问：你发明 Julia 的目的是什么？

Karpinski：在 09 的时候，当我们谈论到技术开发过程中遇到的挫折的时候，主要提到了不同的事情需要不同的语言来做。发明 Julia 的想法就是为了高效。它是一种动态语言，非常简单的编程模型。它有极高的效率。■

本文未完，请看原文：

http://developer.51cto.com/art/201206/341884.htm

内容简介

设计界面的行为不是艺术，而是有规则可寻的。优良的设计界面可以激发、唤起和加强我们与这个世界的联系。

通往优秀UI设计师之路的20个路标

界面设计师 Joshua Porter 在自己的博客中发表了这篇文章，文章中列举了 20 条用户界面的设计原则，这些原则是设计师们在设计工作中需要遵循的，它们能够给设计师们提供较好的指导工作。

界面的存在，促进交互作用

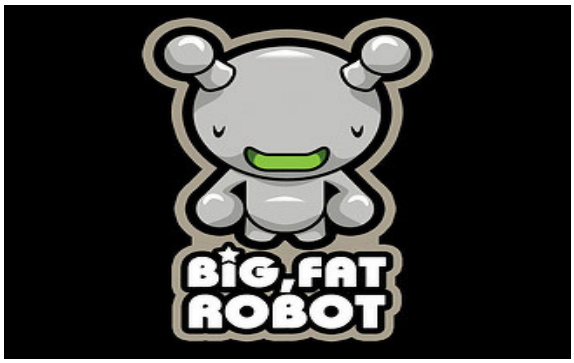
界面的存在，使得用户和我们的世界互动性加强。他们可以帮助用户清晰、阐明、启用等显示关系，它不仅让我们做事有效率，还可以管理我们的应用程序并访问相关的服务。设计界面的行为不是艺术，而是有规则可寻的。优良的设计界面可以激发、唤起和加强我们与这个世界的联系。

清晰度是项很重要的工作

清晰度是界面设计中，第一步也是最重要的工作。要想你设计的界面有效并被人喜欢，你必须要充分认识到界面是什么？人们为什么会使用它？要清楚界面对用户的交互作用，比如当用户使用时，能够预料到发生什么，并成功的与它交互。有的界面设计的不是太清晰虽然能够满足用户一时的需求，但放眼看它混淆的界面是没未来的。清晰的界面能够激起人们信息，并促使人们进一步使用。

不惜一切代价保护用户的注意力

我们生活在一个中断的世界。在日常生活中，总是会有许多事物分散我们的注意力，使得我们很难集中注意力安静地阅读。因此能够吸引注意力是很关键的。所以千万不要将你重要应用周围设计的乱七八糟分散人的注意力……记得屏幕整洁能够吸引注意力的重要性。如果你非要显示广告，那么请你在用户阅读之前显示完。保护和尊重用户的注意力，不仅让用户更快乐，还使得你的广告效果更好。因此要想设计好的界面，保护用户的注意力是先决条件。



保证用户的控制能力

人类往往都对可以掌控自己以及他们环境而感到开心。不考虑他人感受的软件夺走用户的控制力，迫使用户不得不进入计划外的交互，不仅让用户很不舒服，同时也会有意想不到的后果。保证用户的控制能力，让用户自己决定系统状态，稍加引导，我想你会达到你希望的目标。■

本文未完，请看原文：

http://developer.51cto.com/art/201206/340851.htm

内容简介

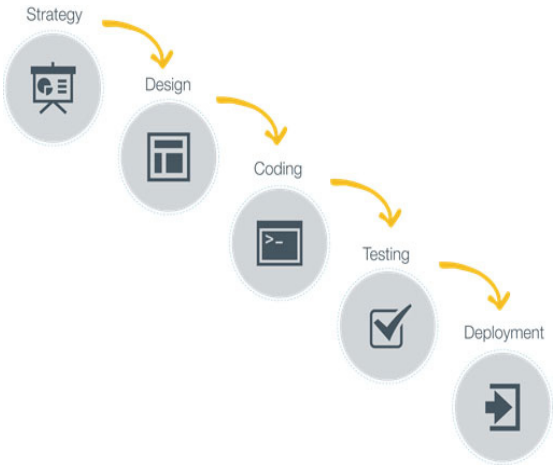
所谓的“ 响应式 Web 设计 ”,它是指在网站建设过程中,利用同一套代码,使网站内容在 PC、平板电脑以及智能手机浏览器上都能正常的显示。那些为了发布同样的内容,而创建 PC, mobile 等多个版本的日子一去不复返了。

5步响应式Web设计和瀑布式说拜拜

你可能熟悉典型的“瀑布模式”的开发过程:从系统需求分析开始,然后着手设计,接着开始前后台开发,最后进行评估并且实施。线性性质是瀑布式开发的主要特点:当这一阶段完成,下一阶段紧接开始,两者配合的几乎天衣无缝。

“瀑布模式”开发过程是通过设计一系列阶段顺序展开的,只需朝一个单一的方向推进工作,而不幸的是,随着问题的不断积累,不得不放慢脚步,来应付各种棘手的问题。

“ 瀑布模式 ”开发过程演示图:



所谓的“响应式 Web 设计”,它是指在网站建设过程中,利用同一套代码,使网站内容在 PC、平板电脑以及智能手机浏览器上都能正常的显示。那些为了发布同样的内容,而创建 PC, mobile 等多个版本的日子一去不复返了。现在你可通过构建一个非常灵活的网站去应付所有的运行环境。

那么,为什么要使用响应式设计而不是采取瀑布模式?瀑布模式只按照标准的桌面浏览器进行设计,除此之外,几乎没有考虑任何其它的设计开发环境,这是它最大的缺点。而敏捷的响应式设计从一开始就考虑到这些跨平台问题,从而进行更详细的前期框架构图、设计和测试,而这些工作恰恰在瀑布模式中被省略了。基于响应式设计的网站一旦完成实施,将正确的呈现在 PC、移动设备和平板电脑上。

既然如此,那么如何在团队中实施响应式 Web 设计呢?下面,我们将回顾典型的瀑布模式的开发步骤并且说明如何使他们变为响应式设计模式。

那么会是哪 5 步响应式 Web 设计和瀑布模式说拜拜呢?下面就将为您一一列举。

第 1 步 :计划

瀑布模式开发

在瀑布模式开发过程中,框架图主要由布局和小部件构成。它们被设定为一个特定的尺寸(通常基于像素),并且几乎没有调整的余地。

这些框架图给出了具体的网格 / 布局的尺寸大小,但是不同的屏幕分辨率会导致布局发生变化,这时一切都变得毫无意义。最终,导航条菜单无法使用,无法进入表单页面,并且界面也会变得凌乱不堪。

5 步响应式 Web 设计和瀑布模式说拜拜 II

响应式 Web 设计

解决此问题并不困难。你需要为不同的视图设计不同的部件，并且不要将一个页面当成一整“页”。页面不是最小的组成单元——而是滚动条、文字内容、表单和其他成份是组成整个页面的最小元素。框架图必须考虑不同的屏幕尺寸，因此布局尺寸也是不固定的。布局可以从三列变至两列，在最小的显示设备上（移动智能手机），甚至可将其调整为单列显示。

同时，你也需要改变网站的用户体验——在小尺寸的屏幕上，要求滚动条不仅仅是可以通过鼠标进行操作，而是人们用手指也能够控制它。这样框架图仅仅是一个原型设计工具，而不再是模板，并且需要通过一些开发和测试来确保其能在显示屏上执行。如果在这些初步测试之前开始设计的话，一些未知的开发问题就会接踵而至。不管怎样，项目根本的愿景必须是保持不变的，因此，保持部门之间开放的沟通渠道是必不可少的。

第 2 步 :设计

瀑布模式开发

在瀑布模式开发中，接下来将按照框架图来进行设计，并且通过字体、颜色、间距以及其他设计工具、手法使其变得丰富多彩而富有生机。通常情况下，设计会进行来回的修改，并且通过设计的不断更新，来逐步完善品牌和设计的规范。

响应式 Web 设计

为了更好的使用分配的项目时间和资源，响应式的 Web 设计应该设计不同尺寸的布局和部件。响应式的 Web 设计不再使用基于像素的完美设计。我们认为在不固定的网格中设计灵活的部件，设计不同尺寸的布局和部件的工作量是可控的，虽然完成可以兼容桌面浏览器的设计就已经极具挑战性了。

让 HTML 在所有环境中采用流体布局来提高设计品质。不去专注于用户的总体体验，而考虑每一种浏览器宽度是非常浪费时间的做法。例如，需要确保在小型移动设备上操作 rotating banner 的原件是快速反应的，并且按照行业推荐的最小的 44px 作为标准的人类手指尖的尺寸来进行设计。对于用户体验的设计和针对所有屏幕尺寸进行外观设计是同等重要的。

第 3 步 :开发

瀑布模式开发

在瀑布式开发方法中，一旦客户确认了设计图，接下来的前端开发中，就会发现在小型屏幕中存在各种问题。不幸的是，由于瀑布模式的线性特征，这些不可预见的问题只能随着项目的推进而出现。

响应式 Web 设计

在敏捷的响应式的开发过程中，设计必须以灵活网格为基础。需要由开发者对部件进行规划和原型设计，并且在每一个阶段都进行测试。为了确保部件是可能的最小的组成单元，需要对代码进行优化。

因为部件可以容易的加入到布局中和从中移除，所以在最初的设计中并没有规划出来。通过开发者、设计师和策划者之间良好的协作来规避由于必需的修改而引起的各种问题。这样，团队中的成员达成共识，就可以早点发现和解决问题。

5 步响应式 Web 设计和瀑布模式说拜拜 III

第 4 步 :耐心的测试

瀑布模式开发

在标准的瀑布模式开发的最后阶段是通过单元测试和功能测试来评估站点。在这个阶段发现的问题,可能会要求重新规划项目最初的设想,甚至有时候一个新上市的设备可能对项目造成沉重的打击。需求分析团队和设计团队不得不重新规划和设计以顾及到这些变化,并且将花费更多的时间来进行修改。

响应式 Web 设计

在响应式开发过程中,每一个阶段都会在多种浏览器和不同尺寸屏幕中进行测试,因此问题可尽早发现。这样也可发现某种移动环境与框架图不匹配的问题,以及了解该设计在不同平台上的性能。响应式 Web 设计会尽早的完成项目原型,让客户更早的检查成果以实现双赢。

第 5 步 :尽早发现问题,尽早解决

瀑布模式开发

传统的瀑布式开发过程中,没有通过设计和界面迭代的过程。瀑布式开发忽略项目建设过程中的小细节,从而引起一些问题并与客户期望相冲突。

虽然通过不断的及时与客户沟通,最终问题解决了,但是这些糟糕决策的严重性却仍未被察觉。

响应式 Web 设计

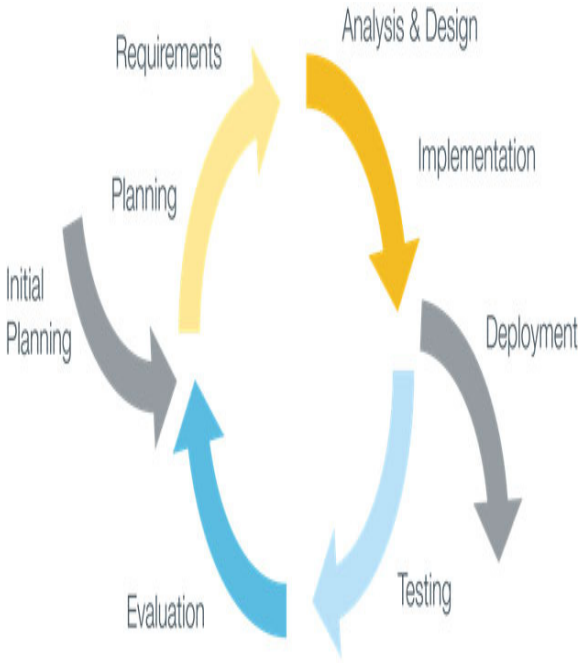
响应式解决方法,在取得同样进展的同时,采用动态代码向客户展示每一步的实现过程。这样,这些早期的工作有利于推动下阶段工作,并且可以在截至日期之前进行关键的修改。

总结

采用敏捷的响应式 Web 设计,可以将你从瀑布模式中解放出来。它将简化你的设计和开发工作,让你的工作更有效率,并在所有可能的平台上最大化宣传你的品牌形象。

真正的挑战是跳出瀑布式开发模式而成为一个响应式设计团队。但只要按照以上的五个步骤去做,你就会与瀑布模式说 “bye, bye”,而对响应式 Web 设计说 “hello”。

“响应式 ”设计过程如下图所示 :



本文作者 Travis Sheppard 是 BGT Partners 的技术副总裁。BGT 在 2010、2011、2012 年被《广告时代》评为 15 个最佳工作地点之一。

它为全球的企业提供互动营销和技术解决方案,以帮助企业加强品牌宣传,挖掘更多的合作伙伴和进行业务变革。■

实时Web时代：都谁玩得起

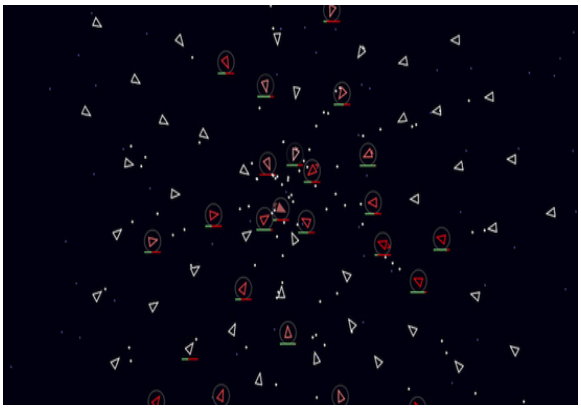
实时 Web 并不只是一种时尚，而是一种技术转移——即时访问 Web。实时技术使 Web 应用变得更快，在某些情况下，几乎与桌面应用没有区别。将来，实时技术将成为一种默认技术，不是只有谷歌、Facebook 和 Twitter 才能玩得起。

北京时间 5 月 18 日消息，国外媒体近日发表文章称，实时 Web 并不只是一种时尚，而是一种技术趋势。将来，实时技术将成为一种默认技术，不是只有谷歌、Facebook 和 Twitter 才能玩得起。

以下为文章内容摘要：

实时 Web 并不只是一种时尚，而是一种技术转移——即时访问 Web。

实时技术使 Web 应用变得更快，在某些情况下，几乎与桌面应用没有区别。



德利斯勒“失败版”MMO《小行星》

实时技术最初的灵感来源很简单——开发经典街机游戏《小行星》(Asteroids) 的在线版本，不同的是可以多人同时在线游戏 (MMO)。

得益于强大的网络后端的支持，在线版《小行星》需要支持数百人同时游戏。更重要的是，要做到实时效果，每个游戏玩家都能毫无延迟地看到每次射击，每个动作。

对于 Hacker news 社区而言，这是一件不幸的事情，因为他们最初发现，这根本无法实现，只是一个愚人节玩笑而已。

不难理解，这让不少游戏玩家感到失望，但是该创意听起来确实可行。随着技术的发展，似乎距离该目标已经不远。例如 Node(用来编写高性能网络服务器的 JavaScript 工具包) 的出现，使得用很少的几台服务器即可同时处理大量用户的指令。此外还有 WebSocket 协议，使得用户之间的持久连接变为可能。

但 JavaScript 开发人员赛博 · 李 – 德利斯勒 (Seb Lee-Delisle) 仍然认为不可行，他认为，一旦真的推出 MMO 版《小行星》，其弹性和稳定性均无法保证，因为游戏要经常交换数据，调整设置，检查冲突等。德利斯勒说：“告诉大家一个不好的消息，如果真的推出 MMO 版《小行星》，那么游戏也将很乏味。”

MMO 版《小行星》成为现实

而另一名开发人员维克拉姆 · 尼扎 (Vikrum Nijjar) 则希望将其作为一个挑战，认为是可行的。幸运的是，尼扎与其他开发人员合作，已经开发出了使之变为现实的软件 Firebase。这一次，尼扎的 MMO 版《小行星》变为了现实，同时也登上了 Hacker News 的首页。

实时 Web 时代 :不只 Google、Twitter 玩得起 II

尽管也略微有些缺陷,但却保证了所有动作的实时显现,做到了最小化延迟。

“实时” 是技术界颇受欢迎的一个流行语,是开发人员、商家和公共关系代表等用来描述一种体验或即时在线互动的。例如,Twitter 被赞许为实时信息源,Facebook 也利用好友的更新和图像实时更新用户首页,一个科技博客网站也承诺对内容进行实时更新。

但是,在上述所谓的实时服务中,都存在延迟,短的几秒钟,长的几分钟。只是我们没在意,或没必要计较而已,因为我们最终获得的内容仍具有价值,让我们满意。

尽管如此,事实上的延迟不可否认。在聊天或发表评论时,这些延迟是可以容忍的。但随着数据的越来越复杂,在某些情况下,如大型多人同时在线游戏,延迟是不可接受的。

今天的许多互联网应用的运行模式是:客户端向服务器发送数据请求,然后从数据库抽取数据。如果数据发生任何变化,应用自身需要再次与服务器进行核实,免得用户手动刷新页面。这种反复连接服务器的过程被称为“轮询” (polling)。

但在实时环境下根本没有轮询,而是订阅 (subscriptions)。一个客户端订阅数据库中的数据,无论何时当数据变化时,客户端都将接到更新。这种变化无需轮询,是因为它们是主动推送的,正如某些通知被推动给智能手机。

15 年出现一次转变

创建富媒体实时应用的 Web 框架 Meteor 联合开发人员马特 · 德波伽里斯 (Matt Debergalis) 称:“我们当前正处在这种转移时期,这种转移通常每 15 年发生一次,所有代码都要重新编写。”

德波伽里斯解释道,在 20 世纪 70 年代至 80 年代初,软件主要运行在大型主机和服务器上,用户使用非智能设备连接。

到了 80 年代后,客户端 – 服务器模式出现,随后互联网诞生。

德波伽里斯说:“大约 15 年后,Web 再次要求我们重写所有软件。因为我们将从桌面应用转向新的模式,这一次软件再次运行在服务器上。”但不同的是,这一次的终端是 Web 浏览器。

如今,桌面软件华丽的用户界面正在消失,取而代之的是一些简单的 Web 功能。从 Gmail 和 Rdio 即可看出这一变化,访问终端是 Web 浏览器。

Rdio 应用

这就是实时连接,应用在外观和表现上与桌面应用几乎没有区别,我们预计将来的互动也没有区别。Rdio 就是一个出色的 Web 应用,在浏览器中运行,与在桌面上运行几乎没有区别。

当然,实时技术并不是什么新鲜事物,低延迟的软件和硬件,以及对信息的几乎同步访问已存在多年。

但开发人员菲尔 · 莱格特 (Phil Leggetter) 称:“没有人在金融以外的领域使用。”

莱格特称,只是最近几年,实时技术才从小众领域走进大众市场。Twitter 是一个转折点,使得信息可以被实时发现。但莱格特称:“下一个目标是实时投递。” ■

未完,查看网络原文:

http://developer.51cto.com/art/201205/338264.htm

HTML 5能够增强Web安全性？

那么，Flash 的消失是否是一个安全利好消息 ?HTML5 是否会替代 Flash? 如果会，那么 HTML5 安全性能否与 Flash 对抗？安全人员应该如何做好部署 HTML5 Web 内容的准备？下面，我们将会针对这些问题展开讨论。

尽管现在所有连接互联网的计算机都安装了 Flash(Adobe 问题不断的 Web 多媒体格式)，但是似乎它很快就会被新标准 HTML5 所替代。按照 Adobe 自己话说，“HTML5 现在得到主流移动设备的普遍支持，并成为创建和部署面向移动平台浏览器内容的最佳解决方案。”

对于企业攻击者而言，这无疑是一个坏消息。近几年来，Flash 已经成为恶意程序黑客的主要攻击目标。根据安全研究公司 WhiteHat Security Inc. 的数据，与 Flash 播放器相关的漏洞占他们发现的 Web 应用程序漏洞的 14% 左右。

那么，Flash 的消失是否是一个安全利好消息 ?HTML5 是否会替代 Flash? 如果会，那么 HTML5 安全性能否与 Flash 对抗？安全人员应该如何做好部署 HTML5 Web 内容的准备？下面，我们将会针对这些问题展开讨论。

HTML5 得到了许多互联网巨头的支持，包括 Facebook、谷歌和 PayPal。事实上，它正在成为将来的互联网视频标准，并且会替代所有非标准格式，如 Flash 和微软的 Silverlight。Flash 是一种二进制的多媒体内容格式，采用面向对象开发语言 ActionScript，需要安装 Adobe 插件。相反，HTML5 是一种开放源码的标记语言，不需要任何插件就能够运行应用程序。删除了视频播放私有插件，也就关上了常见的攻击载体，因为 HTML5 更新是通过浏览器更新实现的，所以它们的更新速度远远比插件快。然而，HTML5 有更多的计算机资源访问权限，包括本地数据存储，从而也成为新的潜在攻击目标。

对于 HTML5，我主要担心的是，开发人员在未完全理解它的新特性及安全机制之前，就匆匆在网站上添加 HTML5 特性。例如，跨域资源共享 (CORS) 使 Web 服务器允许其他域名的网页访问自己的资源。CORS 放宽了同源访问规则 (Same Origin Rule)，这是 Web 浏览器内置的基础安全措施之一。除非开发人员理解 CORS 的工作原理，否则他们很容易做出错误的假设，使攻击者能够访问所共享的内容。HTML5 跨文档消息也有相同的问题。

如果使用正确，它会很安全，但是如何开发者不确保消息来自自己的网站，那么其他网站的恶意代码可能会发送欺骗性流氓消息。基本的安全原则是，来自浏览器的数据都应该视为不可信数据，因此必须进行验证。在 Web 应用程序开发过程中，一定要检查当前的验证过程和过滤器，因为 HTML5 新元素和属性可能会产生一些意外结果。应用程序内置的基于白名单的过滤器确实更具灵活性。

HTML 5 能够替代 Flash 增强 Web 安全性？ II

如果开发者使用技术的方法偏离该技术原来的目标，那么任何技术都可能出现安全漏洞。例如，HTML5 Web 存储标准为开发者提供了一种更灵活方法，可以替代 Cookie 在浏览器上存储数据。当然，存储用户敏感数据存在一定的风险，可能会受到跨站脚本 (XSS) 的攻击，但是有一些网站已经使用这种技术来存储脚本，以提高页面加载速度。例如：为了节省时间和带宽，前面的 Web 服务 Apture 使用一个 localStorage 对象，缓存它的应用程序逻辑代码，但是与这些脚本在同一个域的页面可能存在 XSS 漏洞，可能会被利用，向缓存注入恶意代码。利用 Apture 服务，恶意代码可能会将漏洞变成面向所有域的持久客户端 XSS 攻击。从第三方提取数据或脚本会创建一种隐含的信任关系。开发者必须认识到这种潜在风险，理解如何在内容放到网站之前对内容进行审查。

将一种技术扩展到它原先的适用范围之外，可能会产生其他的错误。HTML5 是一种异步技术，但是开发者可使用 JavaScript 将它变成同步技术。如果一个事务在转到下一个状态之前必须获取一个响应，那么必须仔细检查业务逻辑控制机制，保证事务处理的顺序是否正确，如数据库事务。

安全团队需要使用 WebSocket API，它可以替代浏览器，向 Web 服务器请求最新的数据。服务器会在出现新数据时才发送数据，从而减少服务器与浏览器的流量。但是，WebSocket 可以绕过许多重要的网络安全控制机制，包括传统的数据包头，而防火墙正是通过检查数据包头来阻挡可疑流量的。基于信誉的防御也会受到影响。这样就增加了防火墙进行深度内容检测的负载，因为只有深度内容检测才能够处理 WebSocket 流量，检查流量的内容、结构和用途。所以再说一次，白名单过滤的效率确实会更高一些。

HTML5 标准机构及浏览器厂商已经完全考虑了如何根除某些特定的安全性和保密问题。然而，HTML5 仍然未成为正式的标准，对于那些未掌握编写安全代码的开发者而言，它肯定还不是一种绝对安全的多媒体 Web 开发技术。即使是对于能够编写安全代码的开发者而言，他们仍然需要面对网络欺诈、恶意软件和拒绝服务攻击。使用 HTML5 代码替换网站原来的应用程序是一个很大的改动，总会遇到一些问题。在开始实施之前一定要全面测试恢复过程，而且同时在开始时就要运行一些重要功能。为了更一步防御各种攻击，我推荐将网站升级到 HTTPS。

任何 HTML5 开发都必须进行渗透测试，而且要使用 HTML5 创建复杂的前台，保证它们的运行结果都符合要求。攻击者肯定会测试浏览器厂商实现的新功能和新数据格式，如、及其属性，从中发现可能导致缓冲溢出和其他攻击的编码错误。这意味着安全团队和开发者必须跟进供应商更新，保证尽快更新补丁和修复安全漏洞。

这比以前使用第三方插件技术显然先进了很多。只要开发者投入足够时间，学习如何安全地使用各种新特性，那么安全行业就有望实现更丰富且更安全的互联网。然而，历史表明，这是不可能的，所以我们总是需要实施强有力的边界防御和渗透测试。